APPLYING PATTERN RECOGNITION AND HIGH-TO-LOW RESOLUTION IMAGE MATCHING TECHNIQUES FOR AUTOMATIC RECTIFICATION OF SATELLITE IMAGES

By

AMR ABD-ELRAHMAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2001

ACKNOWLEDGMENTS

I deeply thank my wife, Marwa, for supporting me while I was doing my research and for her great motherhood to my children Ahmed and Yasmine. Ahmed and Yasmine's smiles gave me the power to confront all the daily life obstacles and hope for the future. I owe my mother and my father all the credit for raising and supporting me. I strongly feel their heartly prayers in every and each step I walk.

I would like to thank my committee professors, Scot Smith, Bon Dewitt, Grenville Barnes, Sumi Helal, and Edward Walsh for their significant contribution and guidance to my work. I would like to send a message of appreciation specifically to Dr. Dewitt and Dr. Smith for the effort, time, and support they provided me throughout my stay in the United States. My great thanks go to Dr. Leonard Pearlstine at the Florida Cooperative Fish and Wildlife Research Unit, who supported and provided me with the data used in the research. Dr. Pearlstine's continuous and friendly advice was a great motivation for my research. I also thank Dr. Janice Honeyman who volunteered and played a major role with Dr. Walsh in the aerial imagery acquisition process.

Finally, I would like to thank my country Egypt that provided me with the funds to do my research. I also thank Dr. Mohamed Nassar my, advisor at Ain Shams University, who continuously supported and advised me ever since I finished my B.S. degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

APPLYING PATTERN RECOGNITION AND HIGH_TO_LOW RESOLUTION IMAGE MATCHING TECHNIQUES FOR AUTOMATIC RECTIFICATION OF SATELLITE IMAGES

By

AMR ABD-ELRAHMAN

December, 2001

Chairman: Scot E. Smith
Major Department: Civil and Coastal Engineering Department

Two major objectives were achieved in this research. A technique for rectifying satellite imagery based on matching small-format aerial images, whose coordinates were achieved through navigation sensors, was introduced. A new technique for matching images with large scale differences was also developed.

Small-format digital images were used as a source for ground control points for rectifying satellite imagery. The process involved the determination of the position of the center point of the small-format images using onboard global positioning system (GPS) receiver and aircraft attitude measuring sensors and matching these images with a satellite image. Using this technique, planned flight lines over a satellite scene area provided sufficient control points for rectifying the satellite scene in approximately 5 hours of data acquisition time and 10 hours of processing time.

One of the problems in applying such a technique, which has not received sufficient interest in the image matching literature, was the need to match images that are significantly different in resolution. The ground instantaneous field of view (GIFOV) of the small-format aerial images wa approximately 25 centimeters while the satellite image had a GIFOV of five meters. The developed matching technique modified traditional least squares matching by adding filter parameters, used to filter the high-resolution image to match the frequency content of the low-resolution image, as unknowns in the least squares matching technique.

The results of both traditional and modified matching techniques were evaluated using 30 control points distributed over the satellite scene. The root mean square errors obtained using the results of the modified matching technique were less than those obtained using traditional matching technique results. This indicated the successful implementation of the new modified matching technique. Less than two pixels root mean square error was achieved in both the x and y directions when object space coordinates of the aerial images center points were derived from the navigation sensors. This accuracy was acceptable considering the accuracy of the navigation sensors.

The correlation coefficient resulting from initial correlation matching provided a good measure for predicting the success of the least squares matching process. The satellite image rectification accuracy was tested using different numbers of matched aerial images. Different correlation coefficient thresholds were used to select these images. The results indicated that increasing the number of images led to an increase in the satellite image rectification accuracy. On the other hand, the results showed that adding more images increased the risk of adding images with false matching.

CHAPTER 1
INTRODUCTION

General Description

With the increase in satellite images collected every day, along with the evolution of the latest generation of high-resolution civilian satellite sensors, automatic and efficient techniques for rectifying or geo-referencing these images became a necessity. In addition, the continuous development of multi-sensor technology for aerial mapping, which employs navigation sensors such as the global positioning system (GPS) and inertial navigation system (INS) to provide positional and orientation information of the mapping sensors, led to the use of small-format, digital camera images in a wide range of applications. In these systems, data provided by the navigation sensors enable fast, simple and accurate determination of the exterior orientation parameters of the aerial images.

In this context, this research is devoted to developing a new technique for geo-referencing satellite images by employing small-format aerial digital images, whose exterior orientation parameters are acquired directly by on-board navigation systems. The developed technique will mainly depend on developing a new method for matching images with severe scale differences.

Various matching techniques have been previously developed in which the radiometric image information along with certain similarity measures were used to identify corresponding locations in the matched images. The majority of these techniques were originally developed for matching aerial stereo-pairs. In these kinds of images, large scale differences between features in the matched images is highly unlikely. Most of these methods focus mainly on the geometrical relationship between matched images assuming a simple model (or sometimes no model) for the radiometric differences between the matched images. The model becomes a necessity when matching images with severe scale difference.

On the other hand, image fusion research has focused on the radiometric characteristics of two images in order to obtain an output image with more information than those existing in either of the original images. In order to successfully fuse two images, both images should be co-registered to the same geometrical reference frame. This image registration process is normally conducted in a separate step before the image fusion process. The separation between the image registration and image fusion steps does not provide optimum results for both.

A traditional solution for this problem uses low-pass filters to filter the high frequency component out of the high-resolution image to match the frequency content of the low-resolution image. The high-resolution image is then resampled and matched with the low-resolution one. One of the problems associated with this technique is the difficulty of defining the appropriate low-pass filter with the appropriate parameters. This can introduce many artifacts that affect the accuracy of the matching process. In addition, if either or both images are subjected to distortions, like those introduced by using inappropriate resampling technique or due to certain defects in the imaging system, a simple low-pass filter may not be the best filter to be used.

In this research, the severe difference in resolution between the aerial images and most of the currently used civilian satellite images necessitates the development of a new matching technique, which is another main objective for this research. The accuracy of the georeferencing process is expected to depend significantly on the accuracy of the matching process between the satellite image and the aerial images.

Statement of the Problem

The process of georeferencing a satellite image can be defined as the process of aligning the image with a ground coordinate system. In this process, the geometric errors in the images are mapped through certain geometrical models. The unknown parameters of these models are computed using well-distributed ground control points in the scene area.

Historically, most of the control points were acquired through either ground surveying techniques or by measuring from maps. Typically, ground-surveying techniques are laborious, expensive and subject to availability and accessibility problems. On the other hand, the availability of maps with suitable scale, finding appropriate features that can be used as control points and the required human interface are the major problems in manually acquiring control points from maps.

Accordingly, this research illustrates the use of small-format digital imagery as a source for ground control points for georeferencing satellite imagery. The process involves determination of the center point position of the small-format images using onboard GPS and aircraft attitude measuring sensors and matching these images with the satellite image. Once matching is achieved the small-format image is considered as a control point. Using this technique, planned flight lines over the scene area can provide large number of control points distributed over large remote areas in a short time.

One of the problems in applying such a technique, which did not receive sufficient interest in the image matching literature, is the need to match images that are dramatically different in resolution. For example, in this research, the ground instantaneous field of view (GIFOV) of the small-format aerial images was approximately 20 centimeters while the satellite images had a GIFOV of five meters. In other words, the high-resolution small-format aerial images have much more detail or high-frequency elements than the low-resolution satellite imagery.

The problem of image registeration is usually handled by assuming a geometric model that describes the relationship between the two images. The radiometric similarity between the two images is used to determine the parameters of the assumed geometrical model. Most image registration techniques do not provide an appropriate radiometric model when it comes to registering images with severe difference in resolution.

On the other hand, current image fusion techniques handle the problem of radiometric and geometric similarities between two images independently. Both geometric and radiometric models are applied in two different steps. In addition, most image fusion techniques are applied on images with relatively moderate scale difference. A scale difference of 20:1 as proposed by this research was rarely handled in image fusion literature.

The traditional solution to the problem of registering two images with severe scale difference uses a low-pass filter to filter out the high-frequency component from the high-resolution images, i.e., the aerial images in this case. Then, the aerial images are resampled to the resolution of the low-resolution image, i.e., the satellite images. Determining the characteristics of the optimum low-pass filter and the distortions introduced into the images through the resampling process are impediments in applying this technique.

In this research, a method is developed in which the filtering and resampling process is performed within the matching process. In this technique, both the resampling and the geometrical transformation parameters are considered unknowns and determined through the matching process.

Research Objectives

One of the main objectives of this research is to develop a new matching technique to handle the problem of severe scale difference between the matched images. Although many applications involve matching images with severe resolution, this topic is rarely handled in the literature. In this technique, no previous assumption on the radiometric filter used to filter the low-resolution image is considered. Accordingly, the filter will be considered unknown in a least squares matching process. In other words, the filter parameters that best describe the radiometric relationship between the matched images will be acquired.

The fundamental objective of this research is to introduce a new method for georeferencing satellite images using small-format digital aerial images, whose positions are determined through navigation sensors. In this method, the newly developed matching technique that accounts for the severe difference of resolution between the aerial and satellite images will be used.

This research provides a method for acquiring control points even in remote areas or areas lacking ground control points, maps or even, in case of not enough distinct features that can be used as ground control points, by matching the small-format aerial images with the satellite image. The scale variations between the high-resolution aerial image and the low-resolution satellite images will be taken into consideration within the developed matching technique. The following points will be covered in the research:

1. Develop a matching technique to handle the problem of matching images with severe resolution difference.
2. Discuss traditional matching methods.
3. Test and compare both traditional and developed matching techniques.
4. Study the overall accuracy of the process of georeferencing the satellite image using small-format aerial images.

Scope of the Research

Chapter 2 of this dissertation introduces the different methods used for georeferencing satellite imagery. Traditional area-based matching techniques such as correlation and standard least squares matching are also introduced. In addition, image construction concepts, sampling theory, the convolution theory, Fourier Transform, and image reconstruction concepts are discussed. Different image filtering methods including low-pass filters, band-pass, and band-reject filters are also introduced.

In Chapter 3, a description of the data used in this research with its preparation steps is presented. The experimental approach utilized by this research is also described. Chapter 4 contains a complete description of the modified least squares matching technique. In this chapter, the modified technique is derived using matrix notations. Numerical analysis comments on the derived technique are presented.

Chapter 5 presents the experiments implemented in this research. Variations of the modified least squares matching are analyzed. Satellite image georeferencing results using both traditional and modified matching techniques are introduced. In Chapter 6, the results introduced in Chapter 5 are analyzed and evaluated. The output filter parameters from the modified matching technique are presented together with visual comparison of the filtered images. Chapter 7 summarizes the research procedure, its conclusion and suggests further recommendations for future research.

CHAPTER 2
BACKGROUND

In this chapter, image registration, rectification, area-based matching techniques, sampling theory and image reconstruction methods together with their limitations and role in current research are introduced.

Rectifying Satellite Images

As mentioned previously, image georeferencing or image rectification can be defined as the process of aligning an image with a ground coordinate system. This process involves a spatial transformation from the original image coordinate system known as the image space to a reference coordinate system known as the object space.

Polynomial transformation, projective transformation and differential rectification are three commonly used methods for image rectification (Novak, 1992). All of these methods map the geometric distortions in the image using certain geometrical models. The unknown parameters of these models are computed using well-distributed ground control points in the scene area. For differential rectification, a digital elevation model of the area is also required.

Two approaches can be used for satellite image rectification. The first approach depends mainly on modeling the satellite at its orbit. In these models, the satellite position, attitude, orbit, scan geometry and earth model are used to produce system corrected images (Westin, 1990; Konency et al., 1987). In the second approach, images are accurately rectified using ground control points. Control points with accurately determined object space coordinates are identified on the satellite image and a geometric distortion model is used to map the image space to the object space through a coordinate transformation and resampling process.

Due to the difficulties of obtaining satellite orbit and ephemeris data and the use of complicated software algorithms, image rectification based on ground control points is the most widely used approach. In general, there are three ways for providing control for geo-referencing a satellite image (Sun, 1992): (1) Measuring control points on map; (2) using ground surveying techniques; and (3) transferring control points from aerial photos using photogrammetric methods.

Measuring control points from maps can be performed either manually or automatically by matching the satellite images with digital maps. Manual selection of control points is the most commonly used method. In this method, features from the images are visually matched with the corresponding ones on maps. Road intersections, building corners, etc. are common examples of these types of features. The availability of maps with suitable scale, finding appropriate features that can be used as control points and the required human interface are the major problems in applying this method.

On the other hand, automatic digital correlation between SPOT satellite image and digital maps was first tested in 1988 (Morris et al., 1988). In this technique, candidate features from the digital maps are selected either manually or automatically and matched with the satellite image. This method suffers from the lack of suitable maps and the availability of a sufficient number of features to be taken as control points. Lack of sufficient candidate features is a common problem in this technique, especially with the limitations of the automatic digital correlation process in finding the correct match for all candidate features.

The global positioning system (GPS) and ground surveying techniques have been used to provide accurate geodetic control points for georeferencing satellite images. In this method, candidate control points are identified in the image and their positions measured using GPS or other ground surveying technique. Alternatively, control points can be measured and marked on the ground and then identified in the satellite image. This technique can provide control points to the centimeter level. On the other hand, such a technique suffers from the time and cost problems associated with any ground surveying process.

Control points for satellite image registration can also be acquired by transferring control points from aerial images (Jacobsen, 1988). Digitized aerial photos can be used to provide control points for a SPOT satellite scene by photogrammetric techniques (Sveldow et al., 1976). In this technique, control points are identified on both the aerial photos and SPOT imagery. A bundle adjustment is then performed to compute the ground coordinates for all control points on the aerial photos. These computed coordinates are used as the control points corresponding to specific pixels on the SPOT imagery. The need for ground control points to perform the bundle adjustment on the aerial photos and the complication of the bundle adjustment process form a major limitation to this method.

Over the last decade, airborne kinematic GPS observations in photogrammetry have been used. There are two important reasons to use airborne GPS: (1) to provide accurate flight navigation and (2) to reduce the amount of ground control required for the aerial triangulation adjustment. Utilizing these observations in aerial triangulation has been discussed in Ackermann and Schade (1993), Curry and Schuckman (1993), and Gruen et al. (1993).

The integration between inertial systems and GPS to provide coordinates and attitude of mobile platforms (Li, 1997) facilitated the use of multi-sensors mapping systems. These systems utilize navigation sensors (e.g., GPS and inertial measuring units, IMU) to provide position and orientation of the platform and mapping sensors (Hein, 1989; Schwarz et al., 1993; Schwarz and El-Sheimy, 1996). A wide range of mapping sensors were used such as charge coupled devices (Abd-Elrahman et al., 2000), airborne laser range (Krabil, 1989; Baltsavias, 1999), and hyperspectral scanners (Kruce, 1988; Vane et al., 1993) to provide the mapping information. Using such systems, huge amounts of spatial data can be efficiently collected in a very short time with fairly simple techniques. The simplicity and efficiency of these methods are two of the motivations behind utilizing aerial digital images, whose spatial positions are acquired through GPS and attitude measuring devices to provide control points for georeferencing satellite images, in this research.

From the previous discussion on the three main methods used for providing control points to rectify satellite images, we can conclude that an automatic method that matches spatially georeferenced digital aerial photos can provide a good solution for most of the limitations of these methods. Accordingly, a survey of the current techniques used to automatically match digital images is introduced.

Image Registration and Matching Techniques

Image registration can be defined as the process of aligning two images with each other so that a pixel on both images representing the same area on the ground should coincide with each other. In other words, image registration can more generally be defined as the process of aligning images with a common coordinate system so that corresponding coordinate points in the images correspond to the same physical region of the scene being imaged (Fonseca and Manjunath, 1996).

This process is essential for many applications including image fusion and digital elevation model creation. The basic idea in image to image registration is to find points or features on both images that correspond to the same object on the ground and assume certain mathematical models to map points of one image to corresponding points in the other image. The process of finding corresponding points or features in two images can be reduced to an image matching problem. Many techniques have been developed to match digital images. These techniques can be broadly divided into signal or areas based matching and feature based matching (Lemmens, 1988).

Area-based matching techniques use image gray values to identify conjugate image locations. A matching occurs when there is enough similarity between the gray values and their distribution in the template and reference images. Area-based matching techniques use all the information included in the image as they match gray level values and their distribution in the matched images. These techniques tend to give higher matching accuracy than feature based matching techniques. On the other hand, area based matching techniques are computationally exhaustive due to computing the similarity measures in pixel levels. In addition, area based matching methods are sensitive to geometrical and radiometric variations between the matched images. The normalized cross correlation (Barnea and Silverman, 1972) and the least squares matching (Ackermann, 1983) are well-known examples of area based matching techniques

Feature based image matching involves performing two different tasks. Features from the two images are first extracted, and then corresponding features are matched with each other. Candidate features commonly used in digital imagery include edges, regions, region centers, line intersections, curvature discontinuities, etc. Feature matching algorithms make use of feature attributes such as shape, color and texture. Each feature in one image is compared with potential corresponding features in the other image. A pair of features with similar attributes is accepted as a match.

Scale-space transform techniques were used successfully for feature based image matching. Corresponding features are identified at coarse scale and then refined at finer scales. This helps in studying the matched images' spatial variations at different spatial frequencies. This is especially important when matching images with different resolutions (e.g., SPOT with Landsat Thematic Mapper or satellite image with aerial image). In addition, techniques utilizing coarse to fine scale analysis were proved to operate faster than those which operate on the full resolution images.

Both Laplacian Of Gaussian (LOG) filters (Greenfield, 1991; Schenk et al., 1991) and wavelet transforms techniques (Djamdji et al., 1993) were successfully used in identifying corresponding features in a multi-scale analysis. These two methods were used and compared for automatic generation of digital elevation models from digitized aerial photos (Huang, 1996). Feature based matching techniques are generally fast since they only employ certain features in computing the similarity measures. In the meantime, more work should be done to extract the features and compute the appropriate similarity measures. Multiresolution approaches for image registration are also used with area based matching techniques (Thevenaz et al., 1998). In this time-consuming type of image registration, multiresolution image matching approaches showed great benefits in reducing processing time.

In this research, area-based matching techniques are implemented. Once a matching is achieved, the matching position on the satellite image will be considered a control point position for which the small format aerial image is providing its ground coordinates. Small-format digital aerial images are widely used in applications ranging from management of natural resources to large-scale map compilation (Warner et al., 1996). A similar image matching approach was used to detect positional errors in small format aerial images used to assist satellite image classification (Abd-Elrahman et al., 2001). The task needs high matching accuracy which can be provided using area-based matching techniques. In addition, due to the small ground coverage of the used small-format aerial images, the number of extracted features that can be used in feature based matching techniques are not enough to ensure or achieve matching.

The satellite image and the small-format aerial images used in this research are referred to as the reference and template images respectively. In the following subsections, both normalized cross-correlation and least squares matching techniques used in this research will be discussed. A digital image will be considered a two dimensional discrete light intensity function. Template images with N_T rows and M_T columns will be denoted $T(x_T,y_T)$ and reference images with N_R rows and M_R columns will be $R(x_R,y_R)$ where:

$$0 \le x_T \le N_T, 0 \le y_T \le M_T$$

and

$$0 \le x_R \le N_R, 0 \le y_R \le M_R$$

Each element of the image is called a pixel. The row and column numbers at which this pixel is located determines the position of the pixel while the value associated with the pixel represents the brightness at this position.

Normalized Cross Correlation Matching

In this technique, the correlation coefficient is used as the statistical measure of similarity between the two matched images. A candidate window of the template image, which is typically much smaller in size than the reference image, is statistically compared with windows of the same size in the reference image. The template window moves over the reference image and the correlation coefficient is computed at every new position in the reference window. In case of perfect matching, the result will be another image that has its peak at the pixel where the template matches the reference image. The position on the reference image that gives the highest value of the similarity measure is considered as the matching position. This method is generally useful for matching images which are unaligned by small rigid or affine transformations.

For a template image T and a reference image R, the two-dimensional cross-correlation function for each template window position can be computed as (Sveldow et al., 1976; Rossenfeld and Kak, 1982; Brown, 1992):

$$C(u,v)=\frac{\sum_x\sum_y(T(x,y)-\mu_T)(R(x-u,y-v)-\mu_R)}{\sqrt{\sum_x\sum_y(R(x-u,y-v)-\mu_R)^2\sum_x\sum_y(T(x,y)-\mu_T)^2}} \tag{2-1}$$

where μ_T is the mean of the template image and μ_R is the mean of the reference image over the window region.

We can notice here that the cross-correlation coefficient is normalized since local image intensity would otherwise influence the value. Normalization of the cross-correlation leads to obtain values between –1 and 1 which can easily be evaluated searching for a peak at which the matching occurs. In order to avoid false matching, the computed normalized cross correlation coefficient at the peak is compared with a selected threshold. If the coefficient is less than this threshold, the matching procedure fails. One of the disadvantages of the matching techniques utilizing cross-correlation is their high computational complexity.

In this research, the resampled and oriented small-format digital aerial images are used as template images and the satellite scene is used as the reference image. Perfect matching is not expected due to many factors such as the differences in scale and rotation angle between the digital aerial images (even after orienting and resampling them) and the satellite scene. The objective of this step is to achieve a matching accuracy of few pixels that enables obtaining good initial values for the more accurate technique named least squares matching.

General Least Squares Adjustment

Least squares techniques were used in this research for matching aerial images with a satellite image and for georeferencing the satellite image. General least squares matching was implemented in the modified matching technique. In the traditional matching technique, a special case from the general least squares matching methodology was implemented. In this section, a brief description of the general least squares matching technique is introduced.

Assuming a problem in which a vector of observations L is related to a vector of unknowns X via a mathematical model:

$$F(X, L) = 0 \tag{2-2}$$

if the number of original observations, n, larger than the number of observations n_0 necessary to compute the unknowns, least squares principal can be used to compute a set of adjusted observations with which a unique set of unknowns can be computed. The least squares principal can be described as the process of minimizing the sum of the squares of the differences between the original and the adjusted observations.

The number of degrees of freedom, *r,* in the system equals the number of redundant observations or the difference between the number of original observations and necessary observations. The total number of independent equations that can be written relating the unknowns and the observations is denoted, *c*. In order to be able to apply the least squares adjustment solution, each of these equations should be in a linear form. Non-linear equations should be linearized using Taylor's series expansion before proceeding into the solution. The general form of these equations can be written as:

$$A(L+v)+Bx=d \tag{2-3}$$

where matrix A and B are called the coefficient matrices and represent the partial derivatives of the conditional equations with respect to observations and unknowns respectively. L is the original observations as previously explained. Vectors v and x are correction vectors to the original observations and approximate estimates to the unknowns respectively. Vector d is a column vector of constants. This equation can be rearranged to take the form:

$$Av+Bx=f \tag{2-4}$$

where

$$f=d-AL \tag{2-5}$$

The dimensions of each of these matrices and vectors are important as they determine the complexity of the solution. A complete derivation of the solution to obtain the vector of unknown corrections x is presented in Mikhail (1976). Following this derivation, the unknown corrections vector x can be written as:

$$x=\left[B^tN^{-1}B\right]^{-1}\left[B^tN^{-1}f\right] \tag{2-6}$$

where:

$$N = (AWA^t) \tag{2-7}$$

W is the observations weight matrix. In our image matching case, all observations are assumed to have equal weights and the weight matrix is reduced to the identity matrix.

A special case of the general least squares solution can be derived if the math model can be written in the form:

$$L = F(X) \tag{2-8}$$

In this case, only one observation exists in the left-hand side of each equation while the right-hand side contains only unknowns. The problem complexity in this case is reduced and the vector of unknown corrections can be computed as:

$$x = \left(A^T A\right)^{-1}\left(A^T L\right) \tag{2-9}$$

Least Squares Matching

The least squares matching technique is considered one of the most widely accepted techniques for achieving sub-pixel matching accuracy. In this technique, a geometric relationship between the template and reference images is assumed. Then, a nonlinear optimization algorithm is utilized to minimize the sum of the squares of the gray level differences between the template image $T(x_T,y_T)$ and reference image $R(x_R,y_R)$. The objective of this algorithm is to reposition and/or reshape the template image in a way that the weighted sum of its gray value difference from the reference satellite image is minimized (Rosenholm, 1987; Agouris, 1992). This technique has been widely used and investigated since it emerged back in the mid-1980's (Gruen, 1985; Helava, 1988; Calitzand and Ruther, 1996). A brief description of the least squares matching technique will be given (Stefanidis, 1993; Wolf and Dewitt, 2000):

The equations relating the template and reference images, assuming an affine geometric relationship between the two data sets can be written as:

$$T(x_T, y_T) = h_0 + h_1 R(x_R, y_R) + e(x,y) \tag{2-10}$$

$$x_R = a_0 + a_1 x_T + a_2 y_T \tag{2-11}$$

$$y_R = b_0 + b_1 x_T + b_2 y_T \tag{2-12}$$

where x_T, y_T and x_R, y_R are the respective coordinates in the template and reference images, a_i, b_i; $i = 0,1,2$ are the affine transformation parameters, h_0, h_1 are the calibration factors for the radiometric shift and scale between the template and the reference image, and $e(x,y)$ is the error value.

The unknowns in this mathematical model are the six affine transformation parameters a_i, b_i and the assumed radiometric shift and scale parameters h_0 and h_1 between the template and the reference image. These non-linear equations can be linearized by means of first order Taylor series expansion and reformulated in terms of the differentials of affine transformation coefficient. The linearized form can be expressed as:

$$\begin{aligned} &T(x_T, y_T) - h_1^o R(x_R^o, y_R^o) - h_0^o - e(x,y) = \\ &h_1^o R_x d_{a0} + h_1^o x_R^o R_x d_{a1} + h_1^o y_R^o R_x d_{a2} + \\ &h_1^o R_y d_{b0} + h_1^o x_R^o R_y d_{b1} + h_1^o y_R^o R_y d_{b2} \end{aligned} \tag{2-13}$$

where

$$R_x = \frac{\partial R}{\partial x} = \frac{R(x_R^o + 1, y_R^o) - R(x_R^o - 1, y_R^o)}{2} \tag{2-14}$$

$$R_y = \frac{\partial R}{\partial y} = \frac{R(x_R^o, y_R^o + 1) - R(x_R^o, y_R^o - 1)}{2} \tag{2-15}$$

Note: the superscript zero denotes a current approximation

The above linearized form is solved iteratively. The affine parameters and radiometric shift and scale parameters are assigned initial values, and updated at each iteration. The initial values for the affine shift parameters a_0, b_0 are obtained from the normalized cross correlation method explained in the previous section. The scale and rotation parameters a_1,a_2,b_1,b_2 are calculated using the aircraft attitude and heading parameters recorded at the time of exposure of the template small format aerial image. It should be mentioned here that x^o_R and y^o_R are the location on the reference image computed using the selected initial affine transformation parameters while $R(x^o_R, y^o_R)$ is the gray value of the reference image computed at this location. Similarly, h^o_0 and h^o_1 are the initial values for the radiometric shift and scale parameters between the template and reference image.

The iterative solution proceeds until convergence is reached. Convergence can be assumed when all corrections to the affine parameters and the radiometric corrections are negligible. An equation can be written for each corresponding pair of pixels in the template and reference images giving a number of equations equal to the number of elements in the template image. Assuming there are N_T rows and M_T columns in the template image, the total number of equations that can be formed will be N_TM_T. In each equation, the left hand side consists of one observation, while the right hand side includes unknowns and constants. These equations are called observation equations.

In matrix notation, the linearized observation equations are expressed as:

$$L - e = AX \qquad (2\text{-}16)$$

where L is $N_TM_T x\ 1$ vector of observations. Each element i in L can be written as:

$$L[i] = T(x_T, y_T) - h^o_1\ R(x^o_R, y^o_R) - h^o_0 \qquad (2\text{-}17)$$

The vector of unknowns, x, has a dimension of 8 representing the corrections to the six affine transformation parameters and the two shift and scale radiometric corrections.

$$X^T = [dh_0\ dh_1\ da_0\ da_1\ da_2\ db_0\ db_1\ db_2] \tag{2-18}$$

The A matrix is called the coefficient matrix has a dimension of $N_T M_T x\ 8$. Each row contains 8 values representing the partial derivatives of the equation with respect to the 8 unknowns. A sample row i in A can be expressed as:

$$A[i] = [1,\ R(\overset{o}{x}_R\ ,\ \overset{o}{y}_R),\ \overset{o}{h}_1 R_{x,}\ \overset{o}{h}_1 \overset{o}{x}_R R_{x.}\ \overset{o}{h}_1 \overset{o}{y}_R R_{x,}\ \overset{o}{h}_1 R_y\ ,\ \overset{o}{h}_1 \overset{o}{x}_R R_{y,}\ \overset{o}{h}_1 \overset{o}{y}_R R_y] \tag{2-19}$$

For each new iteration in the solution, a normal equation system of eight equations is formed and solved. The unknowns vector x which contains corrections to the previous parameter set is computed from equation (2-9). It should be noted here that, in each iteration, the x_R, y_R coordinates are computed using the current approximations for the affine transformation parameters. These values are floating number values and do not represent specific pixels in the reference image. A resampling algorithm is used to interpolate for the digital numbers of the reference image at those computed coordinates.

Sampling and Image Reconstruction

A digital image can be looked at as the result of the conversion of continuous space and brightness distribution into a discrete signal. The theoretical aspects of image sampling and image reconstruction are essential when matching images with different resolution. In the following subsections, both the sampling theory and image reconstruction principles will be introduced. However, these concepts can not be clearly introduced without discussing some of the related background topics including the convolution concept, Fourier transform, the impulse and comb functions. Finally, the basic techniques used for image reconstruction is presented.

Convolution

Convolution can be defined, loosely spoken, as a sliding weighted average of one function with another function providing the weights (Theußl, 2000). In one dimension convolution can be represented as follows:

$$h(x) = f * g = \int_{-\infty}^{+\infty} f(y)g(x-y)dy \qquad (2\text{-}20)$$

The resulting *h* function at certain point *x* is obtained by reflecting *f* about its origin and shifting it to point *x*. Then, a point by point product of *g* and the reflected, shifted *f* is performed. The product results are integrated to give the value of the output function h at point x (Castleman, 1995). An example of a convolution is depicted in Figure 2.1. Two box functions convolved result in a tent function.

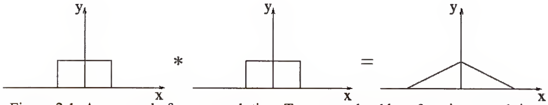


Figure 2.1. An example for a convolution. Two convolved box functions result in a tent function (Theußl, 2000).

The definition of the convolution process can be extended to the two dimensional case as follows:

$$h(x,y) = f * g = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(u,v)g(x-u,y-v)dudv \qquad (2\text{-}21)$$

In this case the value of *h(x,y)* is the volume of the product function of *f* and *g* after reflecting and shifting f to the location (*x*,*y*).

The Fourier Transform

Fourier transform was developed in the Eighteenth century (Bracewell, 1986). The theorem states that it is possible to form any function as a sum of a series of sine and cosine waves of different amplitudes and phases. In this case, the Fourier transform of the function describes the amount of each frequency term that must be added together to make the original function. Functions represented this way, i.e., by describing the frequencies and amplitudes, are depicted in the frequency domain. On the other hand, if a function is defined by values at certain positions, it is represented in the spatial domain.

The spatial and frequency domain representations of a function form two different ways of looking at the function. Some function properties are easily understood in the frequency domain while other properties are clearer in the spatial domain. The conversion from frequency to spatial domain and vice verse is a fully reversible process. To switch from spatial domain to the frequency domain and back we can use the following equations:

$$F(\varpi) = \int_{-\infty}^{+\infty} f(x)e^{-2\pi jwx}dx \tag{2-22}$$

$$f(x) = \int_{-\infty}^{+\infty} F(\omega)e^{+2\pi jwx}d\omega \tag{2-23}$$

The same notation can be extended for two-dimensional functions:

$$F(u,v) = \int_{-\infty}^{+\infty} f(x,y)e^{-2\pi j(ux+vy)}dxdy \tag{2-24}$$

$$f(x,y) = \int_{-\infty}^{+\infty} F(u,v)e^{+2\pi j(ux+vy)}dudv \tag{2-25}$$

Where *u* and *v* represent frequencies in two-dimensions.

Equations (2-22) to (2-25) show that Fourier transform generates complex output from real data input. These real and imaginary parts can be written as:

$$F(\varpi) = \mathrm{Re}(F(\varpi)) + j * \mathrm{Im}(F(\varpi)) \tag{2-26}$$

This also can be written in terms of magnitude and phase as:

$$F(\varpi) = |F(\varpi)| \cdot e^{-j\phi} \tag{2-27}$$

The real and complex parts of the output specify the magnitude and phase of each frequency component, i.e., sine wave. The magnitude, which sometimes is referred to by the frequency response, is defined by:

$$|F(\omega)| = \sqrt{[\mathrm{Re}(F(\omega))]^2 + [\mathrm{Im}(F(\omega))]^2} \tag{2-28}$$

The phase spectrum can be defined as:

$$\Phi(u) = \tan^{-1}\left(\frac{\mathrm{Im}(F(\omega))}{\mathrm{Re}(F(\omega))}\right) \tag{2-29}$$

Often, the Fourier transform is plotted by magnitude versus frequency only and thus ignoring the phase angle. Although this may seem problematic it has become conventional, because the majority of the information of a function is embedded in its frequency content as given by the magnitude spectrum which is also known as the power spectrum (Wolberg, 1990).

It should be mentioned here that the product of two functions in the spatial domain corresponds to the convolution of the functions in the frequency domain and vice verse. This property is very important in understanding some filtering operations. It also has a practical advantage in some cases where it is more efficient to perform a convolution of two functions in the spatial domain by doing multiplication of their transforms in the frequency domain.

Impulse, Impulse Train, and Impulse Lattice Signals

These functions play an important role in understanding sampling and image reconstruction concepts. The impulse function δ(x), which also known as the delta function, is a symbolic function defined by its integral property:

$$\int_{-\infty}^{\infty} \delta(x)dx = 1 \tag{2-30}$$

Technically, the impulse signal is a function which is zero everywhere except at x equals to zero, where it has an infinitely narrow spike with infinite height but it integrates to a value of one (Theußl, 2000). This means that, we can isolate any value from a function f at a value t by shifting the impulse to some value t, multiplying it with f and integrating the result. This property is called the sifting property and can be expressed mathematically as:

$$\int_{-\infty}^{\infty} f(x)\delta(x-t)dx = f(t) \tag{2-31}$$

The impulse train, or comb function, is train of equally spaced impulses

$$comb_T(x) = \sum_{n=-\infty}^{\infty} \delta(x-nT) \tag{2-32}$$

where T is the distance between the impulses. Figure 2.2 depicts the impulse function on the left and an impulse train (a comb function) on the right.

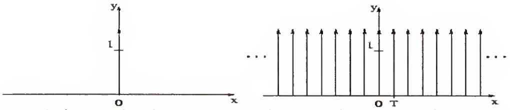


Figure 2.2: The impulse function on the left, an impulse train (comb function) on the right.

In two dimensions, an impulse function $\delta(x,y)$ can be defined by its integral property as:

$$\int_{-\infty}^{\infty} \delta(x, y) dx dy = 1 \tag{2-33}$$

A lattice of equally-spaced infinite impulses in each dimension defines the impulse lattice function as follows:

$$s_T(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(x - m\Delta x)\delta(y - n\Delta y) \tag{2-34}$$

where Δx and Δy are the spacing between the impulses in the x and y directions respectively.

<u>The Sampling Process</u>

In order to sample a continuous function, the function is multiplied by the impulse train (comb) function.

$$f_s(x) = f(x).comb_T(x) \tag{2-35}$$

According to the convolution theorem, multiplication in the spatial domain corresponds to convolution in the frequency domain. The frequency response of the impulse train function is another impulse train function. The spacing between the impulses in the frequency domain equals the reciprocal of the spacing in the spatial domain. Convolving the frequency response of the function with an impulse train causes the function frequency response to be repeated with an origin at every impulse location of the impulse train function. This is shown in Equation (2-36) where a copy of the frequency response of the function $F(x)$ is obtained at each location of the impulse train $comb_{1/T}(x)$. It should be noted here that the spacing of the comb function in the frequency domain equals to the reciprocal of the comb function in the spatial domain.

$$F_s(x) = F(x) * comb_{1/T}(x) \tag{2-36}$$

Figure 2.3 shows how this process replicates the original function in the frequency domain producing replicas located at the positions of the impulses in the impulse train. In this figure, the result of the convolution is many replicas shifted to the positions of the impulses in the impulse train.

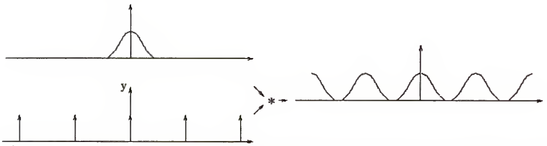


Figure 2.3. Frequency response of some function and an impulse train (left) and the result of convolving these functions(right) (Theußl, 2000).

Having function replicas in the frequency domain leads to major problems when trying to reconstruct the function. If the original function is sampled with large-spacing impulse train, i.e. low sampling rate, the repeated functions in the frequency domain will be very close to each other and may overlap. The possibility of this to happen increases if the function has high frequency components or if as we previously mentioned the function replicas in the frequency domain are very close from each other indicating low sampling rate for the function in the spatial domain. This is clearly shown in Figure 2.4 where the sampling rate was lowered so that the distance between the impulses in the impulse train decreased and the resulting replicas on the right side overlapped.

A digital image can be looked at as the result of sampling a continuous space and brightness distribution. In this case an impulse lattice with fixed spacing in each direction is used. As the case for one dimensional sampling, in order to fully reconstruct a sampled image, the function from which the image is sampled should be band limited and sampled above the Nyquist frequency (Wahl, 1987).

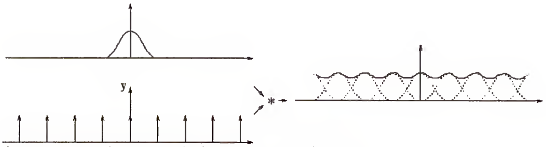


Figure 2.4. Function sampled with a lower sampling rate..

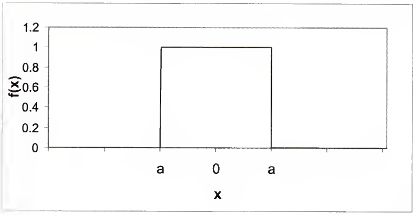
According to the sampling concept, two conditions have to exist in order to be able to completely reconstruct a sampled function. First, the continuous function should be band limited. Second, it should be sampled with a sufficiently high sampling rate to ensure large enough distance between the impulses in the frequency domain so that we don't get overlapped copies of the function in the frequency domain. According to Shanon's sampling theory (Bracewell, 1986), this frequency should be higher than twice the maximum frequency in the function, which is called the Nyquist frequency.

In order to perfectly reconstruct adequately sampled band-limited function, the frequency response of the original function is multiplied by a box function with a half-width equal to the bandwidth of the original function.

$$\Pi_a(x) = \begin{cases} 1 & if \quad |x| \le \frac{a}{2} \\ 0 & if \quad |x| > \frac{a}{2} \end{cases} \tag{2-37}$$

where a denotes the half-width of the box function. This process will result in isolating one copy of the sampled function in the frequency domain. Again, This multiplication process will be a convolution process in the spatial domain. The Fourier transform of a box function is known as the sinc function. Sample box and sinc functions are shown in Figure 2.5. The sinc function can be defined as follows:

$$\text{sinc}(ax) = \frac{\sin(ax)}{ax} \tag{2-38}$$



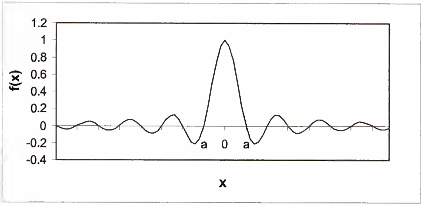


Figure 2.5. The box function (top) and the sinc function (bottom).

In other words, an ideal reconstruction of a band-limited function can be theoretically done either by multiplying the function frequency response by a box function or convolving the function in the spatial domain with a sinc function. Unfortunately, this technique is not practical for many reasons. One of the main reasons is the infinite extension of the sinc function in the spatial domain. In order to use a sinc function practically, it should be trimmed. This changes the nature of the sinc function, which no longer becomes a box function in the frequency domain. Truncating a sinc function means multiplying it with a box function in the spatial domain. Many other finite filters can be used instead of a box function in this case (Press et al, 1988; Theußl, 2000). Generally, we can state that the wider the used filter the better the sinc function approximates a box function in the frequency domain.

Several reconstruction techniques have been used. Three of these methods are the most prevalent, namely (Moik, 1980; Smith et al., 1995): nearest neighborhood, bilinear interpolation and cubic convolution. These three methods will be introduced in the following sections

Nearest Neighbor Interpolation

In this method, the function value at any position is given the value of the nearest sampled point. Nearest-neighbor interpolation can be done by convolving the function with a box function in the spatial domain. The frequency response of the box function is the sinc function, which differs from the ideal box function causing severe aliasing effects.

The Nearest Neighborhood interpolation technique for an image is performed by giving an output pixel the value (digital number) of the spatially nearest pixel in the input image. This method is computationally very simple but it introduces spatial distortions and blocky appearance to the output image (Billingsley et al., 1983).

Linear Interpolation

In this method, the value of the function at any position is given by linearly interpolating between the nearest two sampled points. This method is simple to implement and gives good results. Linear interpolation can be represented by a tent filter. To compare linear interpolation with ideal interpolation in frequency domain we note first that linear interpolation corresponds to convolution with a tent function, which is defined by

$$tent_T(x) = \begin{cases} 1 - \frac{|x|}{T} & if \ |x| < T \\ 0 & else \end{cases} \qquad (2\text{-}39)$$

where T is the sampling distance. The frequency response of the tent function is depicted in Figure 2.6.

Dealing with images, the same concept of linear interpolation is applied. The digital number of the output pixel is found by using linear interpolation scheme using the four nearest pixels surrounding the output pixel in the input image. In two dimensions, this method is known as bilinear interpolation. Resulting images are much smoother and more accurate and the method is computationally more exhaustive than the nearest neighborhood interpolation method.

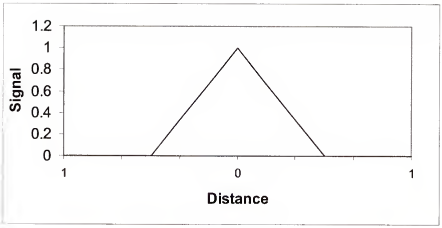
Cubic Convolution Filters

Piecewise cubic polynomials or splines can be used as approximation to the sinc function. Usually, these functions are defined over 4-sampled neighborhood points surrounding the unknown function position (Smith et al., 1995). Many cubic spline functions are proposed (Keys, 1981; Michelle and Netravali, 1988). One of the most commonly used set of spline functions are the ones published in Billingsley (1985). These functioned can be defined as:

$$\begin{aligned} f_1(x) &= (a+2)x^3 - (a+3)x^2 + 1 & 0 \le |x| \le 1 \\ f_2(x) &= ax^3 - 5ax^2 + 8ax - 4a & 1 \le |x| \le 2 \\ f_3(x) &= 0 & |x| > 2 \end{aligned} \qquad (2\text{-}40)$$

where a is a free parameter equal to the slope of the function at $x = 1$.

Generally, $a = -0.5$ gives best results. Figure 2.7 represents these spline functions and their frequency response.



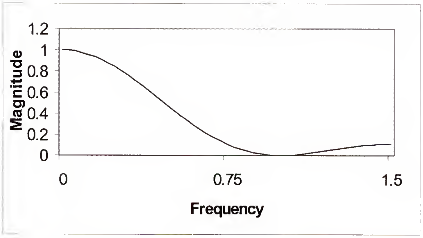
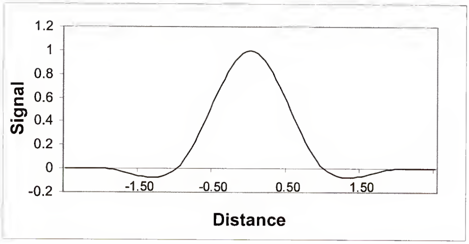


Figure 2.6. A tent function (top) and its frequency response depicted as frequency vs. magnitude(bottom).



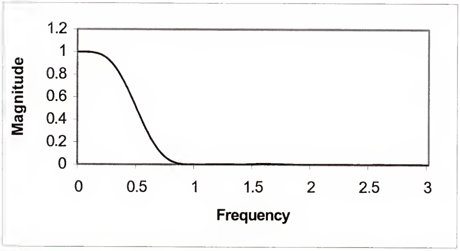


Figure 2.7. Cubic spline functions and their frequency response.

Accuracy and smoothness of the output image can be significantly improved by using the cubic convolution method. In this case, spline functions are used to interpolate between a 4x4 neighborhood surrounding the output pixel in the input image. Although, the cubic convolution resampling technique gives better results in terms of the smoothness and the accuracy of the output image, it still deviates from a box function in the frequency domain introducing aliases and ringing effect to the output image. Moreover, this method adds computational overhead that increases the required computational time.

At this point, we can conclude that image reconstruction can introduce aliasing artifacts or details that are distorted in size and location (Burke, 1996). Small features may appear too large and may be shifted from their true locations. In this case the image information can be so fundamentally altered that it precludes correct visual interpretation (Lorre and Gillespie, 1980). This effect can increase dramatically if large changes in scale need to be applied to the image. This is case when resampling a high-resolution small format aerial to match a much lower resolution satellite image.

One of the solutions to this problem is to use a low-pass filter in order to cut off high frequency components of the high-resolution image before the sampling step. This will reduce the aliasing effect and the distortions expected if the high-resolution images are directly downsampled. One of the most commonly used low pass filters is the Gaussian filter.

<u>Gaussian Low Pass Filter</u>

A one-dimension Gaussian function can be represented as:

$$G(x) = e^{-x^2/2\sigma^2} \tag{2-41}$$

This definition can expanded for two-dimensional Gaussian function as:

$$G(x,y) = e^{-(x^2+y^2)/2\sigma^2} \tag{2-42}$$

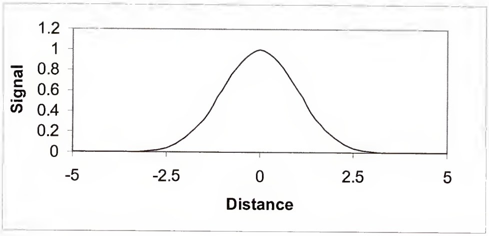
The degree of smoothness of the Gaussian function is controlled by the standard deviation parameter σ. Figure 2.8 shows a one-dimension Gaussian in the spatial and frequency domains.

The convolution of two Gaussian functions is always a smoother Gaussian function. The Gaussian function in two dimensions is a separable function. This means that convolving an image with a two-dimensional Gaussian filter can be done in two steps. First, convolve the image with one dimension Gaussian filter in one direction. Then, the resulting image is convolved with the one dimension gaussian filter in the perpendicular direction. This reduces the complexity of computations for image filtering using Gaussian filters.

Another very important property for the gaussian filter is that the Fourier transform of a Gaussian is also a Gaussian. In other words, the Gaussian function provides a low pass filter with smooth behavior in both the spatial and frequency domain. In this research, Gaussian low pass filter will be used to filter out high frequency elements our of the high resolution aerial images to match the low frequency content of the low resolution satellite image. Different standard deviation values will be experimented to test the image matching techniques with different frequency content.

Band-Pass and Band-Reject Filters

A filter that passes frequencies only in a certain range is called a band-pass filter. Such a filter that can pass frequencies between f_1 and f_2 can be represented mathematically as:



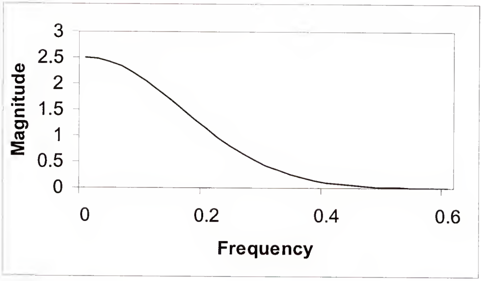


Figure 2.8. Gaussian function and its frequency response.

$$G(s)=\begin{cases}1 & f_1 \le |s| \le f_2 \\ 0 & elsewhere\end{cases} \tag{2-43}$$

In order to, get the representation of this function in the spatial domain, the Fourier transform of this kind of function can be expressed as (Castelman, 1996):

$$g(t)=\Delta s \frac{\sin(\pi \Delta s t)}{\pi \Delta s t} 2\cos(2\pi s_0 t) \tag{2-44}$$

where $\Delta s = f_2 - f_1$, $s_0 = (f_1 + f_2)/2$ and $\Delta s < s_0$. Figure 2.9 shows both the spatial and frequency response representation of an ideal band-pass filter.

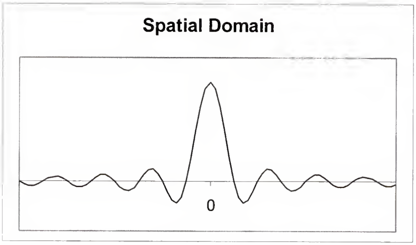
Similarly, a filter that passes energy at all frequencies except for a band between f_1 and f_2 can be represented mathematically as:

$$G(s)=\begin{cases}0 & f_1 \le |s| \le f_2 \\ 1 & elsewhere\end{cases} \tag{2-45}$$

In this case, the spatial representation of this function obtained by deriving its Fourier transform can be expressed as (Castelman, 1996):

$$g(t)=\delta(t)-\Delta s \frac{\sin(\pi \Delta s t)}{\pi \Delta s t} 2\cos(2\pi s_0 t) \tag{2-46}$$

where $\Delta s = f_2 - f_1$, $s_0 = (f_1 + f_2)/2$, $\Delta s < s_0$, and $\delta(t)$ is the delta or the impulse function. The transfer function and the impulse response of this filter are shown in Figure 2.10. This filter is called a notch filter if $\Delta s = f_2 - f_1$ is small.



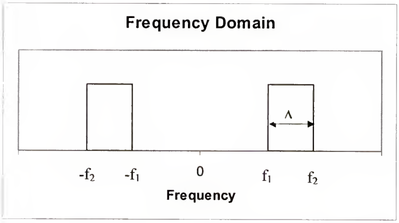
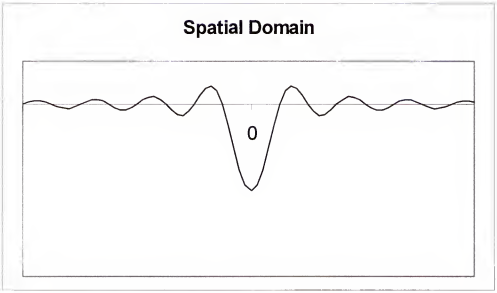


Figure 2.9. Ideal band-pass filter in both spatial (top) and frequency (bottom) domains.



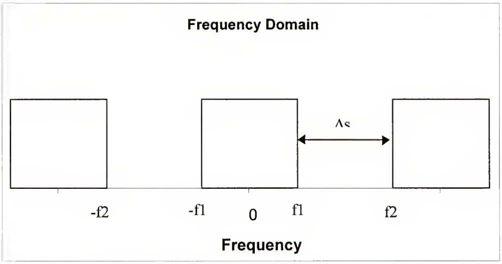


Figure 2.10. Ideal band-reject filter in both spatial (top) and frequency (bottom) domains.

Modeling and Correcting Geometric Distortions in the Satellite Image

Satellite images have numerous sources of geometric distortions. Orbit perturbations, changes in altitude, imperfection in sensor mechanical and optical parts and relief displacement are examples of these sources (Lillesand and Kiefer, 1994; Schowengerdt, 1997). As stated previously, the objective of satellite image georeferencing or rectification is to correct for these geometric distortions and register the image to a ground coordinate system.

Due to the complexity and ambiguity of the characteristics and values of these image distortions, many models have been suggested to correct for them. The most common approach is to assume that these distortions can be described by certain geometric model or mapping function and use image landmarks of known ground coordinates as control points to obtain the geometric model parameters.

Image rectification methods can be divided based on the nature of used geometric models to methods employing polynomial functions, projective transformation and differential rectification models (Novak, 1992). In polynomial rectification methods, the transformation between the original and rectified images is done by polynomials. This method is most often used for satellite images, whose geometry and distortions are difficult to model. Using polynomial mapping functions has the advantage of being easy to implement and provides a simultaneous correction for all sources of satellite image geometric distortions.

A common problem with global polynomial functions is that in case of images with local geometric distortion, the least squares technique, which is used to determine the parameters of the polynomial model, averages the local geometric distortions all over the image. In order to solve this problem, linear and cubic piecewise mapping functions can be used (Goshtasby, 1986; Goshtasby, 1987).

Projective transformation is also used as a mapping function for line scanners satellite images. The well-known point-perspective projective equations derived for aerial images are modified to account for the line-perspective geometry of the line-scanner satellite images. The major problem in this model is its failure to depict the earth curvature distortion. Accordingly, this model has little practical significance for satellite scenes, but could be applied for airborne line scanners.

In this research polynomial mapping functions are used. Polynomial models are simple and have been proved to be creditable especially when stereo images or DEMs are not available. On the other hand, the complication of the differential rectification models and the need to assume a certain model for the changes with time of the satellite attitude angles and the satellite orbit parameters added more advantages to the polynomial approach.

Different polynomial models are used for satellite image georeferencing. Some of these methods assume that the images have only linear geometric differences and geo-reference the images by the transformation of Cartesian coordinate systems (Stockman et al., 1982; Barnea and Silverman, 1972). Affine transformation (Watson et al., 1982) and global polynomial transformations (Wie and Stein, 1977; Lecki, 1980) applied to the entire image were commonly used. The idea behind using affine or polynomial transformation is to assume that the image distortion can be depicted using a certain geometric model. This model ranges from a plane in the case of affine transformation to a higher order polynomial surface. Then the least squares adjustment technique is used to compute the geometric model parameters by minimizing the sum of the squares of the differences between the control point coordinates and the corresponding image coordinates. A general polynomial geometrical model can be represented as:

$$X = \sum_{i=0}^{N} \sum_{j=0}^{N-i} a_{ij} x^i y^j \tag{2-47}$$

$$Y = \sum_{i=0}^{N} \sum_{j=0}^{N-i} b_{ij} x^i y^j \tag{2-48}$$

Where N is the order of the polynomial, a_{ij}, b_{ij} are the polynomial coefficients, X, Y, x, y are the corresponding coordinates of objects in the object space and image space coordinate systems, respectively.

At worst, a quadratic polynomial (2nd order polynomial) is sufficient to describe the distortions in most commercial satellite images if the terrain relief is small and the field of view is not large. The affine transformation is a first-order polynomial that describes shift, scale, sheer and rotation in the satellite image. Affine transformation equations can be written as.

$$X = a_0 + a_1 x + a_2 y \tag{2-49}$$

$$Y = b_0 + b_1 x + b_2 y \tag{2-50}$$

where X, Y are the control point coordinates, and x,y are the corresponding image coordinates. Although using high order polynomials can give an overall better fitting between the two data sets, unexpected and undesired undulations may be introduced.

Statistical Testing

Overall accuracy of the georeferencing process can be evaluated using control points of known ground coordinates. Coordinates for these points can be computed using satellite image georeferencing results. These computed coordinates can be compared with the corresponding known ones to evaluate the accuracy of the matching and georeferencing process.

Root Mean Square Error (RMSE), standard deviation, and mean can be computed for the differences between computed and known ground coordinates of the checkpoints. Different types of statistical tests are performed to test the statistical validity of the satellite image rectification process and to compare the image rectification results obtained from different matching techniques.

The RMSE gives an indication of the overall accuracy of the georeferencing process as it measures both random and systematic errors in the results. In the meantime, the standard deviation value indicates how close the values are from each other and measures the random errors in the results. In addition, the standard deviation of the results of any two-georeferencing methods can be statistically tested to indicate any significant difference between their precisions. The mean values of the differences between the computed and known coordinates can be statistically compared to a population mean of zero indicating if the results are significantly biased from their true values.

Two statistical tests are used in this research namely: the t(student) test and the F-test. The first test is used to test the mean of a sample against the mean of the population. The second test is used to compare the precisions of two samples by testing the equality of their variance. A brief description of both tests is introduced.

Testing Sample Mean against Population Mean

The *t*-student statistical test is conducted on the mean of the discrepancies between the computed and known coordinates in both the x and y directions. The critical *t*-student statistic at a certain confidence level is determined from statistical tables based on the number of degrees of freedom.

The *t*-student statistic is calculated for the x and y discrepancies for each test result using the mean and the standard deviation values using the equation:

$$t = \frac{\bar{x} - \mu_0}{S / \sqrt{n}} \tag{2-51}$$

where μ_0 is the hypothesized population mean ($\mu_0 = 0$ in this case), x is the sample mean, S is the sample standard deviation, and n is the sample size.

The tested null and alternative hypothesis are:

$$\begin{aligned} H_0 &: \bar{x} = \mu_0 \\ H_a &: \bar{x} \neq \mu_0 \end{aligned} \tag{2-52}$$

The null hypothesis can be rejected if and only if the computed *t* statistic exceeds the critical *t* statistic extracted from the tables at the pre-specified confidence level.

<u>Testing the Ratio of Two Population Variances</u>

The F-statistic test is conducted to compare the standard deviation or the variance of discrepancies obtained using the modified and traditional matching techniques. The critical F statistic is extracted from the tables at a certain confidence level knowing the number of degrees of freedom of both samples.

The F-statistic is calculated for the x and y discrepancies using the equation:

$$F = \frac{S_1^2}{S_2^2} \tag{2-53}$$

where S_1 and S_2 are the standard deviation of the two samples and $S_1 > S_2$. The tested null and alternative hypothesis are:

$$H_0 : \frac{S_1^2}{S_2^2} = 1$$
$$H_a : \frac{S_1^2}{S_2^2} > 1 \qquad (2\text{-}54)$$

The computed F statistic for the x and y discrepancies is compared with the critical F statistic extracted from the tables. The null hypothesis is rejected if the computed F value exceeds the critical F value at certain confidence level. This indicates that at certain pre-specified confidence level the variances of the two samples are different from each other.

CHAPTER 3
EXPERIMENTAL APPROACH

Small-format aerial images were acquired and matched with a geometrically uncorrected Indian Remote Sensing satellite image. Once a matching occurred, the matching location was considered a control point for georeferencing the satellite image. For each control point, the ground coordinates of the center point of the aerial images were provided by onboard navigation sensors. A brief description of the data and instruments in addition to the methodology developed for conducting this research are introduced in this chapter.

Description of Data and Instrumentation

The methodology developed in this research was applied on a 5-meter resolution, geometrically uncorrected panchromatic Indian Remote Sensing satellite (IRS) sub-scene. This image was taken by IRS-1D satellite on March 25th, 1998. The image covers an area of approximately 22km x 28km over and surrounding Gainesville city, Florida.

The aerial images utilized by the developed technique were taken by a color-infrared small-format Kodak DCS 420 digital camera. These images were acquired at approximately 600 meters altitude with a 20mm focal-length lens. Each image has 1524 x 1012 pixels with a ground coverage of approximately 400m x 265m.

A Garmin 12-channel GPS receiver measuring C/A code was used to provide real time differentially-corrected coordinates for the aircraft location at each exposure. The real-time differential correction was broadcast from commercial ground control stations.

This system can provide coordinates of the antenna location with errors from 1 to 3 meters. The aircraft attitude was determined by a Watson Industries attitude and heading reference system (AHRS-BA303). This device mainly utilizes three solid state gyros to provide two attitude angles and a third heading angle. An accuracy of 0.5 degrees in attitude angles determination can be achieved using this instrument (Watson Industries, 1995).

The three types of sensors: (1) the digital camera; (2) the GPS receiver; and (3) the attitude measuring devices are linked together through an ACCUPHOTO navigational unit. This unit provides the capability of automatic firing of the camera at predefined flight lines with certain distance intervals. In addition, the ACCUPHOTO unit enables recording the GPS and attitude data at the same time the camera shutter is released.

The positional accuracy of the aerial mage centers on the ground computed using recorded GPS and aircraft attitude information was found to be approximately 15 meters and rarely exceeds 25 meters (Abd-Elrahman et al., 2000). It should be mentioned here that with the advance in mobile mapping technology, a sub-meter accuracy in positioning aerial images could be achieved (Schwarz et al., 1993; Li, 1997). Such mobile mapping systems use differential dual frequency GPS receivers with carrier phase type of observations integrated with precise inertial measuring units (IMU).

Typically, system configuration mainly depends on the purpose for which the system is utilized. For example, a mobile mapping system that provides high positional accuracy might be used in large scale mapping applications or for geo-referencing of high-resolution satellite images. Alternatively, land cover sampling used to assess land cover satellite image classification does not usually need very accurate positional accuracy. The system used in this study was originally developed for the Florida GAP project (Pearlstine et al., 2000) in order to be used as a sampling tool to assist land cover classification. This system is expected to be satisfactory for geo-referencing IRS images to an accuracy of one to two pixels.

Over 250 images were taken in 8 flight lines with this system. An image was automatically taken by the ACCUPHOTO navigation system approximately every 750 meters. Figure 3.1 shows the approximate locations of the captured aerial image center points plotted over the IRS image. In order to produce this figure, an initial shift correction was applied to the IRS image. This was needed to get the aerial image locations and the IRS coordinate system close enough for the purpose of illustrating the distribution of the aerial images over the IRS scene. These images were matched with the IRS image as will be explained later in this chapter.

Methodology

The methodology developed in this research for satellite image georeferencing can be summarized in the following basic steps:

- Acquisition of aerial images and data preparation.
- Developing and testing of the matching technique.
- Correcting geometric distortions in the satellite image and comparing accuracy of different matching techniques.

In the following sections, aerial image acquisition and data preparation steps are presented. The outlines of the developed matching technique together with the experimental approach adopted in this research is introduced. Finally, the step of correcting geometrical distortions in the satellite image is discussed.

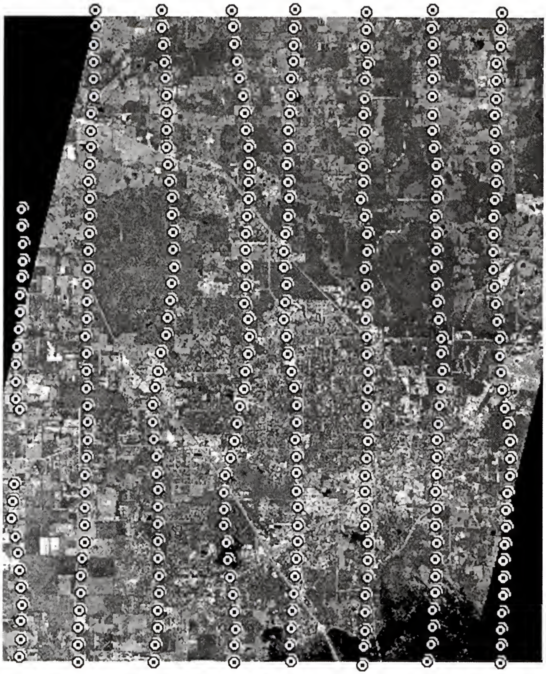


Figure 3.1. Approximate locations of captured aerial image center points plotted over the IRS image.

Image Acquisition and Data Preparation

As mentioned previously, over 250 images were taken in 8 pre-defined flight lines covering the IRS scene area. Figures 3.2 and 3.3 show samples of the aerial digital images used in this research taken over different types of landscape, including urban, suburban, forest, pasture areas. For each one of these images the aircraft location and attitude at the exposure time was recorded. Table 3.1 shows sample data recorded by the ACCUPHOTO navigation system presented in a tabular format.

Table 3.1. Part of tabulated images data file.

Lat.(deg)	Long.(deg)	alt(ft)	GPS head	Date	Time	Roll (deg)	Pitch (deg)	Gyro Head (deg)
29.830192	-82.406610	1922	171	11/21/99	15:23:14	0.5	0.5	164.0
29.823956	-82.405784	1929	174	11/21/99	15:23:28	-1.1	-0.1	165.9
29.817214	-82.404869	1909	173	11/21/99	15:23:43	0.4	-0.7	164.8
29.810539	-82.403892	1850	171	11/21/99	15:23:59	-0.2	-1.4	160.4
29.804000	-82.402445	1774	167	11/21/99	15:24:15	0.0	1.2	156.2
29.797442	-82.400844	1745	169	11/21/99	15:24:32	0.6	-0.1	161.7
29.790829	-82.399909	1748	176	11/21/99	15:24:50	0.3	2.4	169.2
29.784125	-82.399393	1755	174	11/21/99	15:25:10	0.1	3.9	161.7
29.777503	-82.398373	1778	174	11/21/99	15:25:29	1.2	3.8	165.6
29.770775	-82.398097	1794	179	11/21/99	15:25:48	0.8	4.2	172.4

Figure 3.4 shows the aircraft at the time of exposure and the geometry from which the equations to compute the ground coordinates of the center point of the image were derived. These equations can be written as:

$$X_c = X_{GPS} + H' * \tan(\omega) * \sin(\alpha) - H' * \tan(\varphi) * \cos(\alpha) \qquad (3\text{-}1)$$

$$Y_c = Y_{GPS} + H' * \tan(\omega) * \cos(\alpha) + H' * \tan(\varphi) * \sin(\alpha) \qquad (3\text{-}2)$$

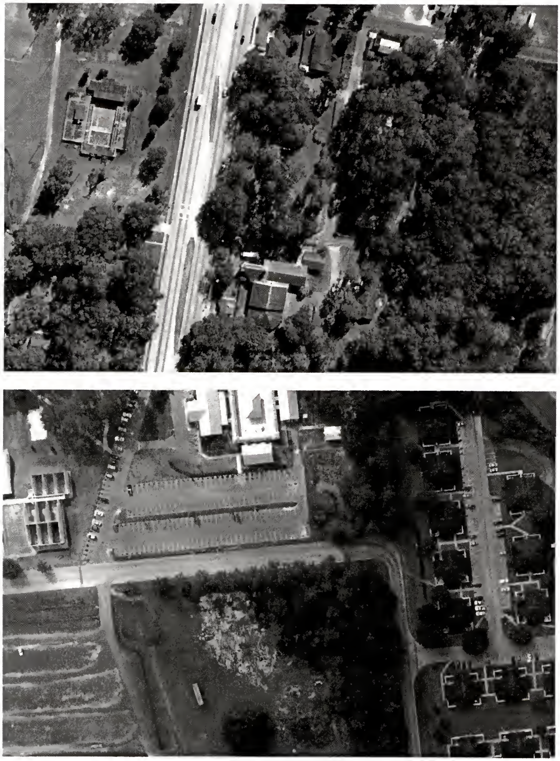


Figure 3.2. Sample small-format aerial images over suburban landscapes.

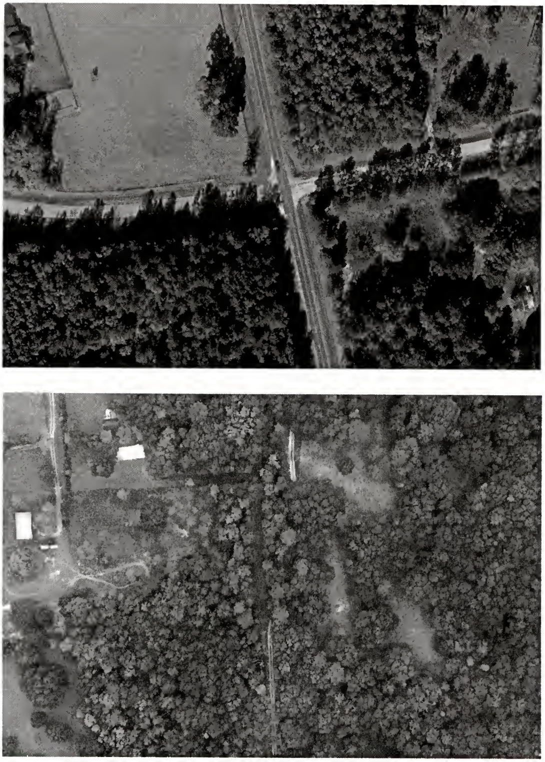


Figure 3.3. Sample small-format aerial images over rural areas.

where:

X_{GPS} : UTM x coordinates of the camera acquired by the onboard GPS receiver.

Y_{GPS} : UTM y coordinates of the camera acquired by the onboard GPS receiver.

X_c : UTM x coordinates of the image center point.

Y_c : UTM y coordinates of the image center point.

H' : The altitude of the aircraft above ground.

ω : The pitch angle of the aircraft at exposure recorder by the attitude measuring device.

φ : The roll angle of the aircraft at exposure recorder by the attitude measuring device.

α : The heading of the aircraft at exposure recorder by the attitude measuring device.

Before leaving this section, it should be mentioned that in all captured images, a blank strip in the right hand side of the images and another blank area in the upper left corner of the images were eliminated. These blank areas showed in the images due to a problem in the camera mount. Although, these blank areas were eliminated from the images before processing, the pixel coordinates of the center of the images were calculated based on the original size of the images. Figure 3-5 shows one of the images before and after eliminating the blank areas.

Developing and Testing Matching Technique

Area-based matching techniques were used in this research. The size of the small-format aerial images after resampling to the IRS satellite image resolution does not provide enough features for feature based image registration. The size of these resampled images can also be practically utilized in area-based matching algorithms. In addition, area-based matching techniques promise higher matching accuracy, which is needed for accurate georeferencing of satellite images.

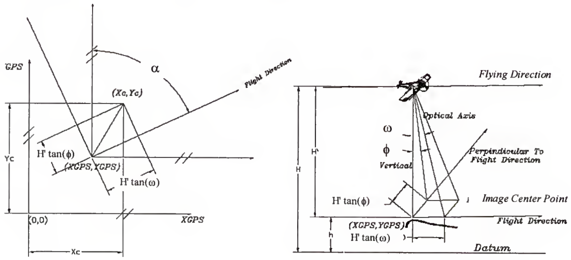


Figure 3.4. Coordinates of image center points from GPS and attitude information.

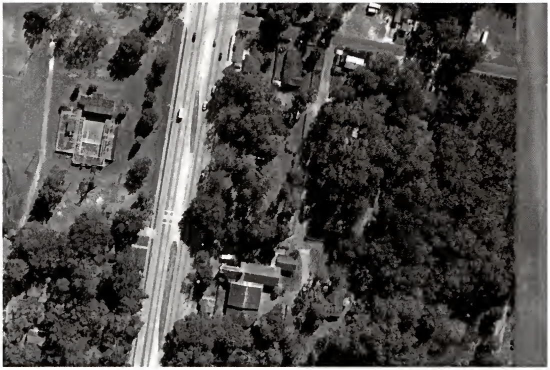
One of the typical problems associated with area-based techniques is their sensitivity to scale difference and rotation angle between matched images. This problem is critical in this research knowing that the captured small-format aerial images have a ground resolution of approximately 25cm, whereas the IRS satellite image resolution is 5m. Moreover, small-format aerial images are not precisely aligned with the satellite image, which results in orientation differences between both types of images. Fortunately, knowing the flight height and the average elevation of the terrain at the image area can provide the average photo scale and an estimate of the scale ratio between the IRS image and the aerial images.

In order to use area-based matching techniques, such as correlation and least squares matching, the aerial images should be filtered using low-pass filter to get rid of high-frequency components (Schowengerdt, 1997), then resampled using one of the resampling techniques discussed earlier in Chapter 2. In addition to the numerous types of low-pass filters that can be used, different filter parameters can be chosen to control the amount of high-frequency component filtered out of the aerial images.

One of the most commonly used low-pass filters is the Gaussian filter. Variations of Gaussian filters can be achieved by changing the standard deviation (sigma) parameter in the used filter. This filter was discussed in Chapter 2 in both spatial and frequency domains. Generally, increasing the sigma value of a Gaussian filter increases the number of high-frequency components filtered out of the images and causes them to be more blurred.

Both correlation and least squares matching techniques used in this research, are significantly affected by the quality of the filtered and resampled aerial images. Choosing a low-pass filter that cuts off the right high-frequency component from the high-resolution images to be similar to the low-resolution image is a challenging problem.

In this research, the traditional least squares matching technique was modified to account for the severe difference in scale between the aerial images and the IRS image. Both the traditional and the modified matching techniques are discussed in the following subsections.



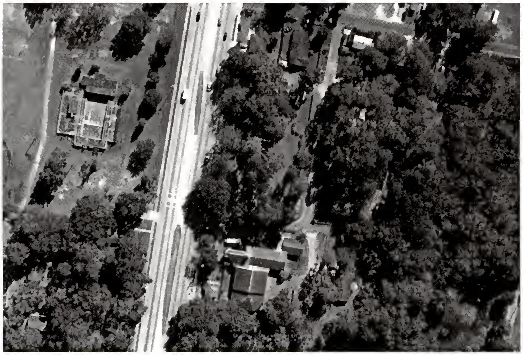


Figure 3-5. Sample aerial image before (top) and after (bottom) eliminating the defected blank areas.

Variations of Gaussian filters were tested using different sigma values of 5,7,9, and 11. The images were then resampled using the cubic convolution method discussed in Chapter 2. Images filtered with a Gaussian filter with sigma values less than 5 and resampled to approximately the same resolution of the IRS image were visually found to have severe aliasing artifacts. On the other hand, images filtered with a Gaussian filter where sigma exceeds 11 were found to be over-smoothed so that many prominent features were diminished and/or eliminated. Figure 3.7 shows the one-dimension Gaussian filter with values of 5, 7, 9, and 11 for the Gaussian filter sigma.

Initial matching was achieved using correlation matching technique. The airplane heading recorded by the onboard navigation sensor was used to account for the orientation difference between the aerial images and the IRS image. Both matching results from the correlation matching and the recorded orientation angles provided approximate values for the affine transformation parameters between each aerial image and the IRS image. These values were essential for the following step of applying the least squares matching procedure.

Modified Matching Technique

Traditional least squares matching assumes approximately no scale difference between the template and the reference images which necessitates filtering and resampling of the high-resolution image to approximately match the scale of the low-resolution image. This is usually accompanied by both spatial and radiometric distortions in the resampled image. In addition, there is always the problem of choosing the correct low-pass filter and the best resampling technique.

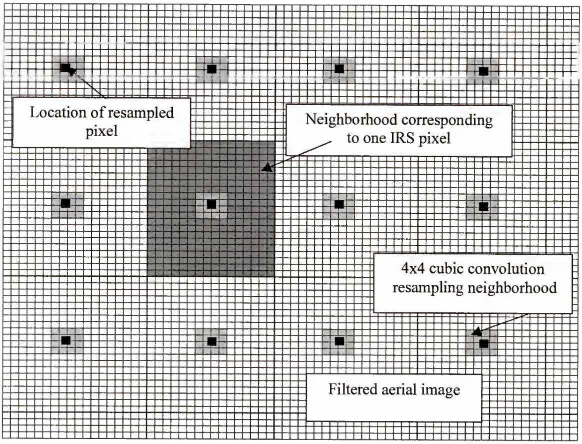

Location of resampled pixel
Neighborhood corresponding to one IRS pixel
4x4 cubic convolution resampling neighborhood
Filtered aerial image

Figure 3.6. Resampling filtered small-format aerial image.

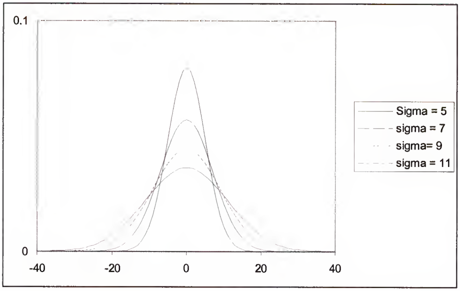


Figure 3.7. Used Gaussian filter with different sigma values.

Therefore, in the proposed modified least squares matching, filter parameters used to filter the high-frequency component from the high-resolution image were added as unknowns in the least squares matching process. This means that the matching process can provide not only the geometrical transformation parameters between the matched images as in the traditional least squares matching, but also the best filter parameters to change the scale of the high-resolution image to match the scale of the lower resolution one. A detailed description of the modified least squares matching technique is introduced in Chapter 4. Chapter 5 emphasizes the tests conducted using both traditional and modified matching techniques.

<u>Correcting Geometric Distortions in the Satellite Image.</u>

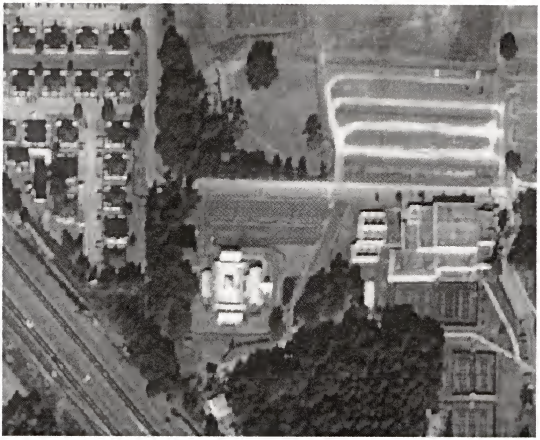
One of the objectives of this study is to provide a method in which a newly developed matching technique can be used for georeferencing a geometrically uncorrected IRS satellite image. In order to achieve this goal, the new method needs to be proven superior to traditional matching methods. Then, using the results of the newly developed matching technique, together with data acquired by the navigation system, the IRS satellite image is georeferenced and the results are tested using known checkpoints.

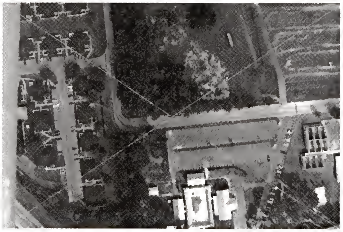
Control points provided by the matched small-format aerial images were used to correct for the geometrical errors in the IRS image. Each control point maps a column and row position on the satellite image to the *x* and *y* ground coordinate system. The satellite image row and column numbers that correspond to the center point of each matched aerial image was considered a control point where corresponding ground coordinates of the center point of the aerial image are known. The geometrical distortions in the satellite image were modeled through an affine transformation model.

Ground coordinates of the aerial images center points were computed from navigation sensors data recorded with each aerial image. An alternative method to get ground coordinates of the aerial images center points is to visually identify these center points on another georeferenced image with reasonable resolution and accuracy. The USGS one-meter resolution Digital Orthophoto Quarter Quadrangles (DOQQ) were considered reasonable candidates for this process. Figures 3.8 and 3.9 show samples of using the DOQQ images to obtain the ground coordinates of the small-format aerial images center point coordinates.

The positional accuracy of the DOQQ which promised to meet the US national map accuracy standards for scale 1:12,000 and their high-resolution, provided better determination of the ground coordinates of the aerial images center points than those determined by the used navigation sensors data. Moreover, a study conducted in areas located within the IRS scene, showed that a root mean square error for the differences between the coordinates of point located on the DOQQ and measured using ground differential GPS were 1.70 and 2.50 meters in the Northern and Eastern directions, respectively, (Orndorff, 1997). The more accurate ground coordinates provided through the DOQQ were needed to compare the accuracy of the traditional and modified matching techniques, which expected to be in a sub-pixel range. This accuracy makes it hard to compare their results unless we have accurate ground control points.

Checkpoints distributed all over the IRS image were selected. Most of these checkpoints represent road intersections, which can be accurately identified on the satellite image. The ground coordinates of seven of these points were obtained using GPS observations (Buhrke, 1991). The ground coordinates of the rest of the checkpoints were obtained by locating these points on the DOQQ to define their ground *x* and *y* coordinates. Figures 3.10 and 3.11 show samples of checkpoints whose ground coordinates were obtained by locating the points on both the IRS and DOQQ images. Two sample checkpoints whose coordinates were obtained through GPS observations are shown in Figures 3.12. Figure 3.13 shows the distribution of all the ground checkpoints plotted on the IRS image. Table 3.2 lists the ground coordinates of these checkpoints.





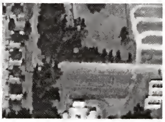


Figure 3.8. Sample of using DOQQ image (top) to obtain ground coordinates of center point of small-format aerial image number 1227-056 (bottom left). Image in bottom right represents a zoom in the DOQQ image representing the center of the small-format aerial image.

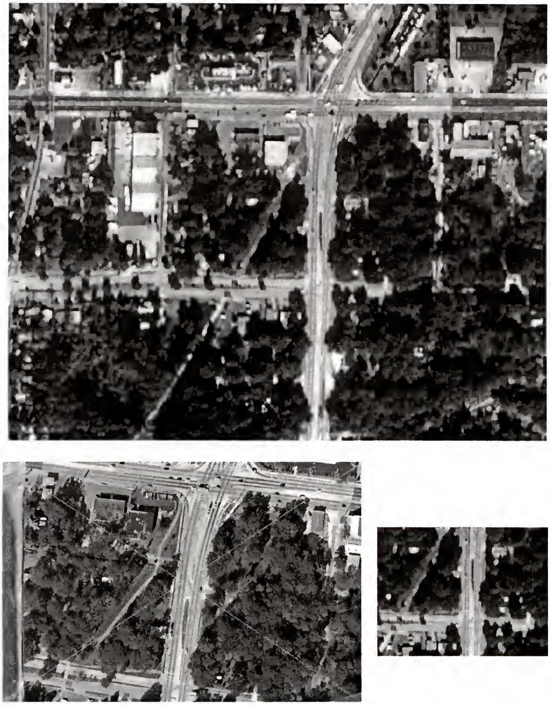


Figure 3.9. Sample of using DOQQ image (top) to obtain ground coordinates of center point of small-format aerial image number 1227-051 (bottom left). Image in bottom right represents a zoom in the DOQQ image representing the center of the small-format aerial image.

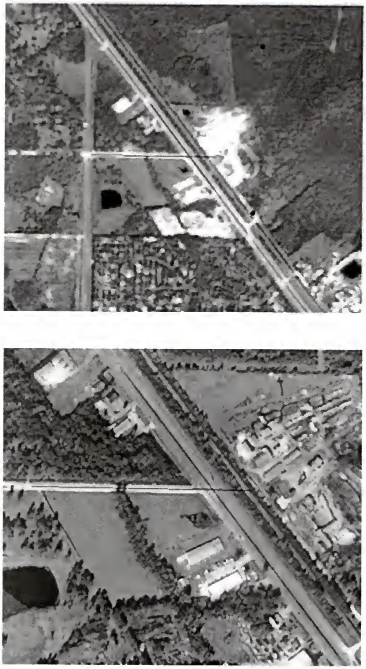
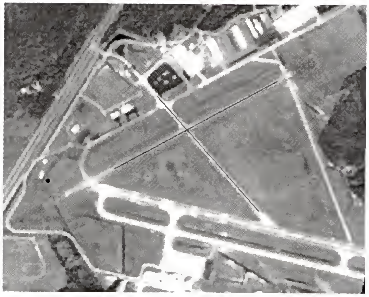


Figure 3.10. Checkpoint DOQQchk-06 located on both the IRS (top) and DOQQ images (bottom).



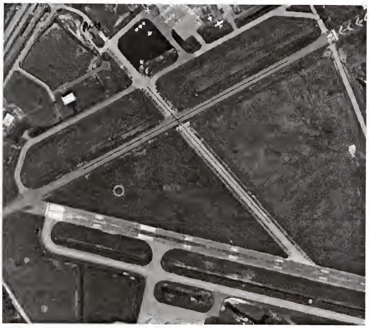
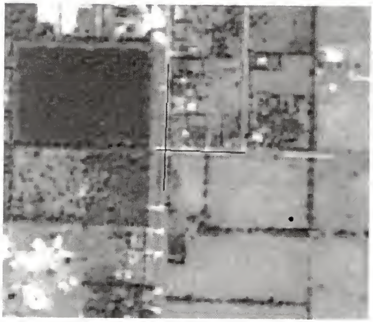


Figure 3.11. Checkpoint DOQQchk-10 located on both the IRS (top) and DOQQ image (bottom).



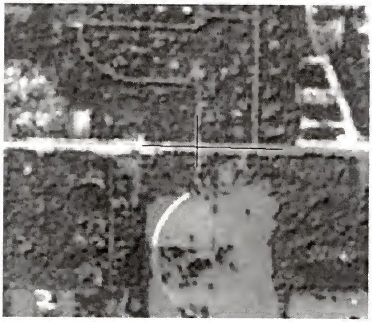


Figure 3.12. Checkpoints GPSchk-05 (top) and GPSchk-06 (bottom) whose coordinates obtained through GPS observations.

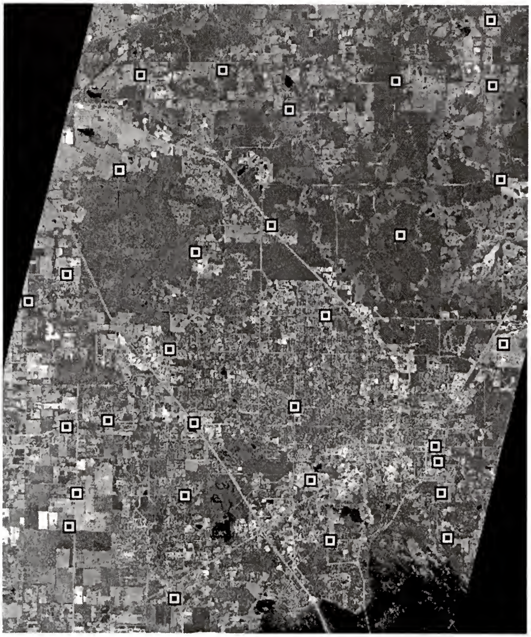


Figure 3.13. Distribution of all checkpoints plotted on the IRS image

Table 3.2. Ground and image coordinates of checkpoints.

Check Point	x(pix)	y(pix)	x(meter)	y(meter)
GPSchk05	1010.3	1233.3	357439.09	3278782.12
GPSchk06	1292.6	1876.2	358857.10	3281993.82
GPSchk07	578.8	2928.3	355285.68	3287250.24
GPSchk08	4330.2	846.7	374033.58	3276880.62
GPSchk11	3279.2	819.9	368795.33	3276734.24
GPSchk18	3234.9	2809.9	368566.53	3286683.35
GPSchk21	3112.0	1355.7	367946.31	3279411.75
DOQQchk01	926.6	1818.4	357020.80	3281704.08
DOQQchk02	951.4	940.5	357146.35	3277322.91
DOQQchk03	4275.8	1241.0	373769.71	3278852.13
DOQQchk04	4242.6	1521.2	373598.47	3280249.37
DOQQchk05	4219.0	1657.6	373477.04	3280933.75
DOQQchk06	2752.4	3614.8	366151.42	3290701.27
DOQQchk07	2056.4	1857.8	362672.50	3281913.57
DOQQchk08	919.9	3174.4	356990.44	3288482.52
DOQQchk09	4828.2	2566.3	376522.68	3285479.47
DOQQchk10	1887.8	308.3	361828.41	3274168.24
DOQQchk11	1979.3	1213.5	362291.12	3278694.09
DOQQchk12	2070.0	3369.5	362738.02	3289471.99
DOQQchk13	2960.0	2004.2	367190.00	3282649.60
DOQQchk14	3910.2	3527.6	371934.62	3290277.10
DOQQchk15	1844.6	2508.9	361615.93	3285160.99
DOQQchk16	2305.3	4978.6	363910.91	3297510.58
DOQQchk17	3860.0	4883.3	371685.43	3297051.69
DOQQchk18	1387.4	4092.8	359334.55	3293077.65
DOQQchk19	4795.9	4025.2	376365.53	3292771.89
DOQQchk20	2902.1	4626.2	366895.22	3295755.18
DOQQchk21	4707.0	5429.4	375919.05	3299787.23
DOQQchk22	4724.4	4847.3	376002.16	3296875.16
DOQQchk23	1578.6	4931.4	360284.90	3297271.37

CHAPTER 4
DEVELOPMENT OF MODIFIED MATCHING TECHNIQUE

The modified least squares matching technique developed to account for large resolution differences between the matched images are discussed here. The hardware and software used and developed in this research are also described.

Modified Least Squares Solution

In the proposed modification, a reconstruction filter for the high-resolution image is assumed. This filter is discretized into a window of elements from which a new digital number is calculated from the high-resolution image to correspond to one pixel in the low-resolution image. The technique considers these window elements as unknowns in the least squares matching process. This means that the matching process results not only in the transformation parameters between the matched images as in the traditional least squares matching process, but also the best filter to account for the scale and the spectral differences between the two matched images.

Recalling the notations used in Chapter 2, a template image with N_T rows and M_T columns is denoted $T(x_T, y_T)$ while a reference image with N_R rows and M_R columns is denoted $R(x_R, y_R)$ where:

$0 \le x_T < N_T, 0 \le y_T < M_T$, and

$0 \le x_R < N_R, 0 \le y_R < M_R$

Each pixel in the reference image corresponds to an $n \times n$ square window $W(k,l)$ of pixels in the template image. An integer m is also defined where:

$0 \le k < n, 0 \le 1 < n$, and

$n = 2m+1,\ m > 0$

Figure 4.1 shows the distribution of the unknown filter parameters in the assumed window. Figure 4.2 shows a diagrammatic representation of the reference, and template images, and the unknown window parameters. Generally, the process starts from the top left corner of the image and proceeds from top to bottom and left to right.

In this case, the equations relating the high resolution template and the lower resolution reference images, assuming an affine geometrical relationship between the two data sets are in the following form:

$$\sum_{i=-m}^{m} \sum_{j=-m}^{m} T(x_L + i, y_L + j) W(i+m, j+m) - R(x_R, y_R) - h_0 - e(x,y) = 0 \qquad \text{(4-1)}$$

where:

$$x_R = a_0 + a_1 x_T + a_2 y_T \qquad \text{(4-2)}$$

$$y_R = b_0 + b_1 x_T + b_2 y_T \qquad \text{(4-3)}$$

x_T, y_T are the coordinates of the location in the template image where the *nxn* window surrounding this location corresponds to a single pixel in the reference image. The coordinates of this location in the reference image is x_R, y_R. The parameters a_i, b_i where $i = 0, 1, 2$ are the affine transformation parameters, h_0 is the calibration factor for the radiometric shift between the template and the reference images, and $e(x,y)$ is the error value. In addition to the six affine transformation parameters a_i, b_i and the assumed shift parameter h_0 between the template and the reference image, the *nxn* filter window elements $W(k,l)$ are added as unknowns to the system.

w(0,0)	w(0,1)	.		.	w(0,l)	.				.	w(0,n-1)
w(1,0)	w(0,2)	.		.	w(1,l)	.				.	w(1,n-1)
w(2,0)	.				.					.	.
.	.	.			.			.			.
			.								
				.							
.	.				.			.			.
.	.				.					.	.
w(k,0)	w(k,1)	.		.	w(k,l)	.		.		.	w(k,n-1)
.					.						.
.					.						.
.											
.	.		.		.						
.	.	.			.			.			.
w(n-2,0)	w(n-2,1)	.		.	w(n-2,l)	.				.	.
w(n-1,0)	w(n-1,1)	.		.	w(n-1,l)	.				.	w(n-1,n-1)

Figure 4.1. Unknown filter window parameters for nxn window size.

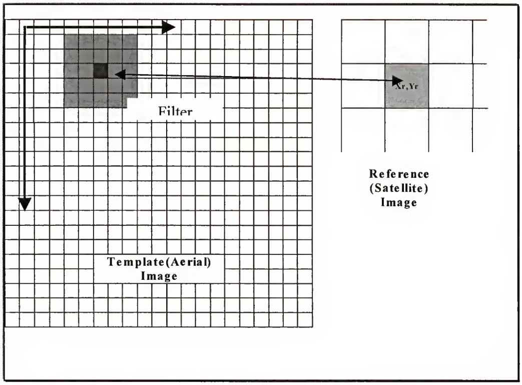


Figure 4.2. Diagrammatic representation of reference image, template image and filter window.

These non-linear equations can be linearized by means of first order Taylor series expansion and reformulated in terms of the differentials of affine transformation coefficient. The linearized form can be expressed as:

$$\begin{aligned}
&\sum_{i=-m}^{m}\sum_{j=-m}^{m}[T(x_L+i,y_L+j)W^o(i+m,j+m)-R^o(x_R,y_R)-h_0^o] \\
&+\sum_{i=-m}^{m}\sum_{j=-m}^{m}[T(x_L+i,y_L+j)dW(i+m,j+m)] \\
&-\left[R_x da_0 + x_R^o R_x da_1 + y_R^o R_x da_2 + R_y db_0 + x_R^o R_y db_1 + y_R^o R_y db_2\right] \\
&+\sum_{i=-m}^{m}\sum_{j=-m}^{m}[W(i+m,j+m)dT(x_L+i,y_L+j)]=0
\end{aligned} \tag{4-4}$$

where:

$$\begin{aligned}
R_x &= \frac{\partial R}{\partial x} = \frac{R(x_R^o+1,y_R^o)-R(x_R^o-1,y_R^o)}{2} \\
R_y &= \frac{\partial R}{\partial y} = \frac{R(x_R^o,y_R^o+1)-R(x_R^o,y_R^o-1)}{2}
\end{aligned} \tag{4-5}$$

The above linearized form is solved iteratively. All the unknown parameters are assigned initial values which, are updated at each iteration. As mentioned in Chapter 2, initial values for the affine shift parameters a_0, b_0 were obtained from the normalized cross correlation matching step. The initial scale and rotation parameters a_1,a_2,b_1,b_2 are calculated using the aircraft attitude and heading parameters recorded at the time of exposure of the template small format aerial image. These values can provide a reasonable approximation to the actual affine transformation parameters between the template and reference images. It should be mentioned here that $\overset{o}{x}_R$ and $\overset{o}{y}_R$ are the location on the reference image computed using the selected initial affine transformation parameters while $R(x^o_R, y^o_R)$ is the gray value of the reference image computed at these locations.

The initial value for the radiometric shift parameter between the template and reference images h^o_0 can be assumed 0. Initial values for the window elements $W(k,l)$ can be taken as $1/n^2$. This means each of the n x n pixels in the template image contributes equally in the equation. The iterative solution proceeds until convergence is reached. Convergence is assumed when all corrections to the unknown parameters are negligible.

The number of equations depends on the size of the template image and the scale between the template and the reference images. An equation is written for each set of *nxn* pixels in the template image corresponding to one pixel in the reference image. Assuming there are N_T rows and M_T columns in the template image, the total number of equations that can be formed is denoted *c* where *c* is an integer value computed from:

$$c = floor(N_T / n) * floor(M_T / n) \tag{4-6}$$

The floor function in the equation operates on a float number. This number is rounded down to the nearest integer value. Figure 4.3 shows, the distribution of the filter window over the template aerial image and how an equation is written for each *nxn* pixels. In the example of Figure 4.3, the computed value of c is 12. This figure represents the simplest case, where there is no overlap between filter windows.

In matrix notation, Equation (4-4) can be expressed as:

$$f + Av + Bx = 0 \tag{4-7}$$

where:

f is a c x 1 vector in which each element can be written as:

$$f[i] = -\left[\sum_{i=-m}^{m} \sum_{j=-m}^{m} [T(x_L + i, y_L + j) W^o(i+m, j+m) - R^o(x_R, y_R) - h_0^o] \right] \tag{4-8}$$

The vector of unknown *x* has a dimension of *(n^2+7) x1* representing the corrections for each of the filter window elements in addition to the six affine transformation parameters and the shift radiometric correction.

$$X^T = [dh_0\ da_0\ da_1\ da_2\ db_0\ db_1\ db_2\ dw_{0,0}\ dw_{1,0}\ dw_{2,0} \ldots\ dw_{n-1,n-1}] \tag{4-9}$$

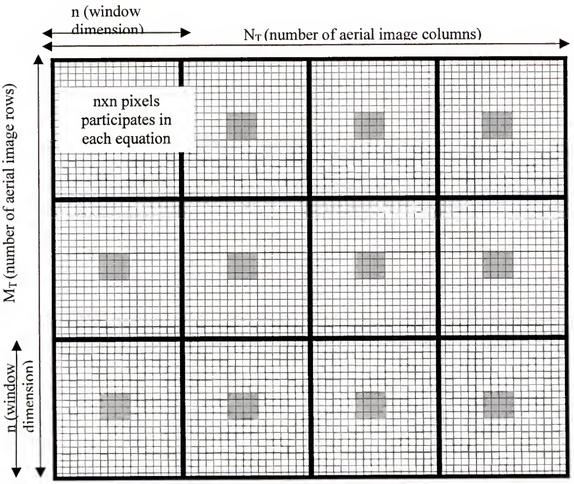

n (window dimension)
N$_T$ (number of aerial image columns)
M$_T$ (number of aerial image rows)
nxn pixels participates in each equation
n (window dimension)

Figure 4.3. Forming an equation for each nxn aerial image pixels.

Array *v* is an $N_T M_T x\ 1$ vector of residuals or corrections to the observations. Each gray value in the template image used in Equation (4-4) is considered an observation. Elements of the vector *v* are corrections to these values.

The *A* and *B* matrices are called the coefficient matrices. The *A* matrix has a dimension of $c\ x\ N_T M_T$. This represents the partial derivatives in Equation (4-4) with respect to each of the observations. A typical row in the *A* matrix can be written as:

$$A[i] = \left[0\ 0\ 0\ .\ .\ .\ 0\ 0\ 0\ W[0][0]\ W[0][1]\ .\ .\ .\ W[n-1][n-1]\ 0\ 0\ 0\ .\ .\ .\ 0\ 0\ 0\right] \qquad (4\text{-}10)$$

When an observation or a pixel in the template image is not participating in the equation the partial derivative of the equation with respect to this pixel is equal to 0. This means that each row in the equation will have only n^2 non-zero elements. Each of these non-zero elements is equal to the value of one of the filter window elements. This is shown in Equation (4-10) representing a typical single row of the *A* matrix. Elements in this row represent the partial derivatives of a single conditional equation with respect to each of the observations, in this case the template matching image pixels participating in the equation. All elements in this row equal to zero except *nxn* elements corresponding to the template image pixels involved in the equation. The values of the non-zero elements equal the window filter parameter.

Figure 4.4 gives a diagrammatic representation of the *A* matrix. In the figure, each row represents the partial derivatives of an equation with respect to the template image digital numbers. The equations are written assuming the filter is moving from top to bottom and from left to right. Each hatched block in a row represents a column in the filter window. Each row in the *A* matrix contains n hatched blocks or *nxn* total non-zero elements. We can notice here that the equations can be written with different overlap between one moving filter window and the next. This overlap is represented by the overlap between elements in two rows in the *A* matrix.

On the other hand, the matrix B has a dimension of c x (n^2+7). Each row contains the partial derivatives of Equation (4-4) with respect to the unknowns. A typical row in *B* can be expressed as:

$$B[i] = \big[1 \; R^o(x_R, y_R) \; R_x \; x_R^o R_x \; y_R^o R_x \; R_y \; x_R^o R_y \; y_R^o R_y \\ T(x_L - m, y_L - m) \; T(x_L - m, y_L - m+1) \ldots \\ \ldots T(x_L + m, y_L + m - 1) \; T(x_L + m, y_L + m) \;\big] \quad . \tag{4-11}$$

The first seven elements of the matrix *B* represents the partial derivatives of each conditional equation with respect to radiometric shift unknown and to each of the six unknown affine transformation parameters. The rest of the elements in the row represent the partial derivative of the equation with respect to each of the *nxn* filter parameters. The values of these elements are the digital number of the template image pixels corresponding to the filter parameters.

The unknowns vector, *x*, is obtained by solving a normal equation system of linear algebraic equations. This linear algebraic system and the derivation of its solution vector were described in Chapter 2. The solution vector for the equations was presented in Chapter 2 and can be computed from Equations (2-6) and (2-7) using the derived *A*, *B*, and *f* matrices.

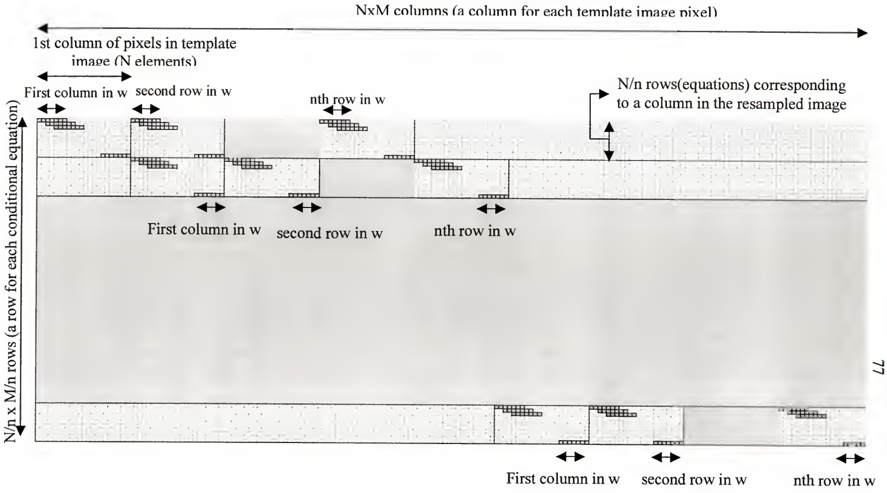

NxM columns (a column for each template image pixel)
1st column of pixels in template image (N elements)
First column in w
second row in w
nth row in w
N/n rows(equations) corresponding to a column in the resampled image
N/n x M/n rows (a row for each conditional equation)
First column in w
second row in w
nth row in w
First column in w
second row in w
nth row in w

Figure 4.4. A Diagrammatic representation of a typical *A* matrix.

Solving for *x* involves computing the inverse of the *N* matrix in Equation (2-6). The *N* matrix is a square matrix with number of rows and columns equals to *c*, i.e. the number of conditional equations in the system. Assuming a 1512x1024 aerial image with a 25cm ground resolution is matched with a 5 meter resolution satellite image, the number of expected conditional equations is approximately equal to 3800 equations.

Computing the inverse of the *N* matrix with this size is a problem knowing that such an inverse needs to be computed for each iteration in the least squares process and for each matched image. However, The *N* matrix is a band limited matrix where its band size depends on the unknown filter window size. This fact would help reduce the computational cost if the purpose was to solve a (*c* x *c*) normal equation system. Unfortunately, in our case, to solve for the unknown vector *x*, we need to compute the inverse of the matrix *N* rather than solving a linear equation system. This means that a full computational cost for matrix inversion is needed for each iteration in the solution.

One way to overcome this problem is to assume a window size so that there is no overlap between the pixels. This will ensure that a pixel from the template image only participates in one and only one of the conditional equations. In this case, the *A* matrix can be written so that all pixels participating in each conditional equation are adjacent to each other. Accordingly, each row in the *A* matrix will have n^2 adjacent non-zero elements. Each element corresponds to one of the unknown filter parameters.

Figure 4.5 shows a diagrammatic representation of the *A* matrix. In this figure, each row has *nxn* adjacent non-zero elements representing the *nxn* elements of the filter window. These elements are the partial derivative of each conditional equation with respect to the template image pixels. According to the assumption of no-overlap between filter window, a unique set of adjacent pixels can participate in each conditional equation. Accordingly, hatched blocks in each row of the *A* matrix are adjacent to each other. In addition, no overlap exists between the hatched blocks in any two rows of the matrix. This assumption will bring the N matrix not only to an easily inverted diagonal matrix but also to a scalar matrix where its inverse is the reciprocal of the scalar value. This by no means a significant improvement in the computational cost for solving the system.

The solution also involves solving the linear algebraic system shown in Equation (4-12) whose size equal to the number of the unknowns in the solution. Increasing the number of unknowns in the model increases the computation cost associated with each iteration in the least squares solution. Considering the *nxn* window parameters as unknowns in addition to the six affine transformation parameters and the radiometric shift unknown, the total number of unknowns in the system is n^2+7. This means that the size of the normal equation system increases with the square of the window size. Different models can be tested in which expected symmetry in distribution of the filter window parameters are used to significantly reduce the number of unknowns in the model. These variations are explained in the next section. The testing experiments and results of the modified matching technique including these filter parameters variations are discussed in Chapters 6 and 7.

Variations of Modified Matching Technique

Different variations of the modified least square matching technique are proposed. In these variations, different assumptions of the distribution of the unknown window parameters were suggested. These variations can be summarized as:

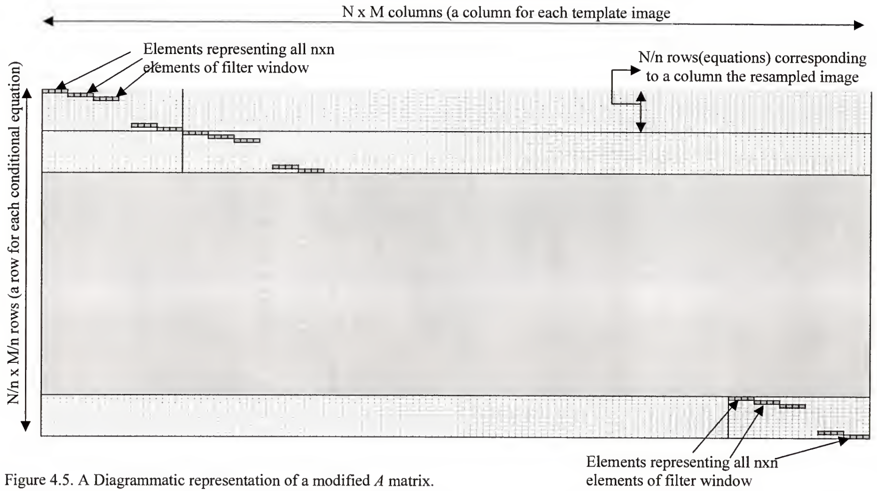

N x M columns (a column for each template image
Elements representing all nxn elements of filter window
N/n rows(equations) corresponding to a column the resampled image
N/n x M/n rows (a row for each conditional equation)
Elements representing all nxn elements of filter window

Figure 4.5. A Diagrammatic representation of a modified *A* matrix.

1. All window parameters were considered unknowns.
2. A four-quarter symmetry was assumed. This means that only the parameters of one quarter of the window were considered unknowns. The rest of the window parameters were obtained through symmetry.
3. Diagonal-symmetry was assumed. In this case only half of the parameters on the diagonal of the filter window were considered unknowns. The rest of the window parameters were obtained through symmetry.
4. Using circular window with diagonal-symmetry. In this case, a circular window was assumed instead of the square window assumed in the third variation.

In the following each of these variations is introduced. Factors concerning the practicality, robustness and adequacy of each method are also discussed.

<u>All window parameters are unknowns.</u>

In this assumption, all the nxn parameters in the filter window were assumed unknowns in the least squares solution. The distribution of the unknown window parameters in this case was shown in Figure 4.1. The number of unknowns in the least squares matching is $n^2 + 7$ which includes n^2 window parameters unknowns, six affine transformation parameters and one radiometric shift unknown. As shown in Equation (4-12), in order to solve for each iteration in the least squares solution, a normal equation system of $n^2 + 7$ linear algebraic equations should be solved. Assuming a window size of 21x21 elements, a set of 448 linear equations needs to be solved. Solving a normal equation system with this size is a slow process especially when dealing with several iterations for each of the matched images.

Four-quarter symmetry

In this variation, a four-quarter symmetry in the filter window parameters was considered. This assumption can be supported by the nature of the image acquisition system. Figure 4.6 shows the distribution of the unknown window parameters in this case. Again, considering a 21x21 filter size example, this variation reduces the number of filter parameter unknowns to 121. Therefore, the size of the normal equation system that needs to be solved for each least squares iteration was reduced to 128 equations, which represents a significant reduction in the computational cost of solving the normal equation system.

w(0,0)	w(0,1)	.	.		.	w(0,n/2+1)	.		.	.	w(0,1)	w(0,0)
w(1,0)	w(1,1)	.	.		.	.	.		.	.	w(1,1)	w(1,0)
w(2,0)	.	.						.			.	w(2,0)
.	.		.							.	.	.
.												.
.		..		..		.				.		.
.					.	.					.	.
w(n/2+1,0)	.	.		.	.	w(n/2+1,n/2+1)	.	.		.	.	w(n/2+1,0)
.	.										.	.
.		.								.		.
.												.
.	.										.	
w(2,0)	.					.					.	w(2,0)
w(1,0)	w(1,1)	.			.	.	.		.	.	w(1,1)	w(1,0)
w(0,0)	w(0,1)	.			.	w(0,n/2+1)	.		.	.	w(0,1)	w(0,0)

Figure 4.6. Unknown filter window parameters for four-quarter symmetry case.

Using Diagonal Symmetry

The number of window filter parameters was reduced further by assuming that only half the parameters that lie along the diagonal of the filter window are unknowns.

The rest of the window parameters were computed through linear interpolation based on the distance from the center of the window assuming full rotational symmetry. This reduced the number of window unknowns to 11 if we are using a 21x21 filter window size. Figure 4.7 shows the distribution of these diagonal unknowns.

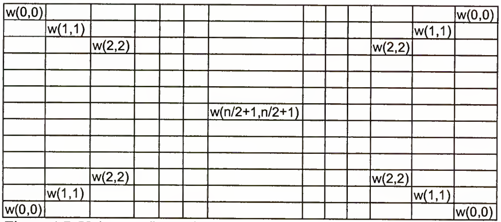


Figure 4.7. Unknown filter window parameters for Diagonal-symmetry case. Only the diagonal elements assumed unknowns. The rest of the elements (non-diagonal blank spaces) are calculated using linear interpolation assuming a full rotational symmetry.

The size of the normal equation system that needs to be solved in each step of the least squares solution is reduced to 18 equations. Solving linear algebraic equation system of 18 equations was considered very reasonable in terms of computational cost.

Using Diagonal Symmetry with Circular window

In this case diagonal-symmetry was assumed. Window parameters were considered equal to zero if the distance between the parameter location and the window center exceeds half the window size. Such configuration eliminates the unknown parameters at the window corners and approximates the filter window to a circular shape.

The number of unknowns and the size of the normal equation system were the same as in the diagonal-symmetry assumption. The purpose of this variation was to study how the use of a circular window affects the matching results.

All the introduced variations for the modified least squares matching techniques involve solving linear algebraic equations. One of the most important steps in solving this kind of equations is to analyze the robustness of the obtained solution. In the following section, a brief discussion of the singular value decomposition analysis technique, utilized in this research, is introduced.

Singular Value Decomposition Analysis

This type of analysis was performed to detect and handle singular or ill-conditioned sets of linear equations. Failure to achieve a unique solution for the unknowns can happen when one or more of the equations is a linear combination of the others. In this case the equations are called singular (Press, et al., 1988). In some cases, the equations are not singular, but the round off errors in the machine bring them closer to singularity and the numerical procedure might fail. Accumulated round off errors in the solution can produce a wrong solution. This happens particularly if the number of unknowns is large.

The increased number of unknowns with the probability of having some highly correlated unknowns in the developed least squares matching technique raises the possibility of obtaining a singular or ill-conditioned set of linear equations. Singular value decomposition (SVD) is the most widely used technique for analyzing the condition of linear algebraic equation systems, especially those involved in least squares procedures. SVD can not only diagnose the condition of the equations but also, in some cases, solve the singularity problem. A linear algebraic equation system of the form:

$$Ax = B \tag{4-12}$$

where A is the matrix of coefficients, x the vector of unknowns, and B is the right hand side vector can be written in the following format:

$$(UwV^{T})\,x = B \tag{4-13}$$

In this form matrix A is decomposed into 3 matrices, U, w, and V^T. The most important matrix to diagnose the condition of the system is the w matrix which is a positive or zero elements diagonal matrix in case where A is a square matrix. The condition number of the matrix A or the linear algebraic system is the ratio between the largest and smallest elements of the w matrix. If this value is infinite, the matrix is considered singular while it is considered ill-conditioned if the reciprocal of the condition number approaches the floating-point precision of the machine which was approximetly 10^{-15} for double precession computations.

Singular value decomposition analysis was done on the linear algebraic equations of the least squares solutions for some of the aerial images using MATLAB software. Linear algebraic matrices were imported into MATLAB and the condition value for the matrices were computed. This was done only for sample images to make sure that ingeneral, adding the filter window parameters as unknowns in the least squares solution does not numerically lead to singular or ill-posed linear algebraic systems, which in turn fails or weakens the solution.

Computer Hardware and Software

Analysis conducted in this research was mostly performed using a PentiumII PC machine operating under Windows 98 operating system. IBM workstations with the AIX operating system were used to run ERDAS IMAGINE software. The software used in this research was mainly developed by the author. Different commercial software packages were also used in the data preparation, results testing and analysis stages. In the following a brief description of the software developed and used in this research is introduced.

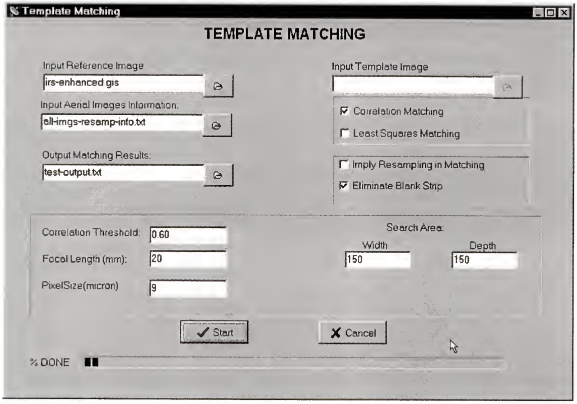
Developed Software

Borland C++ builder version 5.0 software package was used to write the code and Graphical User Interface (GUI) used in this research. An image class was written to perform several image tasks including reading, displaying, and saving images. The major task in this class was to do image matching. Several image matching algorithms were implemented, including traditional correlation and least squares matching. The modified least squares matching algorithm was implemented using the different variations discussed in Chapter 5. It should be mentioned here that the code used for solving linear algebraic system of equations was implemented and modified from the book "Numerical Recipe in C" based on the Cholesky decomposition algorithm (Press, 1988).

Small modules were also written to read the ERDAS format, perform gaussian filtering, and perform bilinear and cubic convolution image resampling techniques. The 'Template Matching' GUI was used to input matched image data, and to select the matching method. Figure 4.8 shows the Template Matching GUI. Appendix C lists the code developed in this research for different image matching and filtering tasks.

Commercial Packages

Several commercial software packages were employed in this research. MATLAB software was used largely for singular value decomposition analysis and to analyze filters in frequency domain. Both matrix and discrete Fourier transform functions were widely used. ERDAS IMAGINE software was used for reading the satellite image raw data, experimenting with image quality and change image formats. Both ARCVIEW and ADOBE PHOTOSHOP software were used to identify small format aerial image center points, enhance images for viewing, and testing matching results. Some Avenue scripts were written and used with the ARCVIEW script to facilitate automatic recording and displaying of control point coordinates.


Template Matching

TEMPLATE MATCHING

Input Reference Image: irs-enhanced.gis

Input Aerial Images Information: all-imgs-resamp-info.txt

Output Matching Results: test-output.txt

Input Template Image:

☑ Correlation Matching

☐ Least Squares Matching

☐ Imply Resampling in Matching

☑ Eliminate Blank Strip

Correlation Threshold: 0.60

Focal Length (mm): 20

PixelSize(micron) 9

Search Area: Width 150 Depth 150

Figure 4.8. Graphical user interface for TEMPLATE MATCHING application.

CHAPTER 5
EXPERIMENTS AND RESULTS

Experiments conducted using the data and the methodology described in Chapter 3 are introduced. The new matching technique developed in Chapter 4 is tested. The results of georeferencing the IRS satellite image using traditional and modified matching techniques are presented.

Testing the Modified Matching Technique

Each one of the small-format aerial images was matched with the IRS image. The matching process was conducted in two stages. First, correlation matching was performed to provide initial values for the least squares matching solution. Then, the modified least squares technique described in Chapter 4 was used. The correlation coefficient computed within the correlation matching step provided an indication of the degree of similarity between the matched images. High correlation coefficient values promise more robust results from the following least squares matching technique.

Each of the four variations for the modified matching techniques was considered. These variations can be recalled from Chapter 4 as:

1. All window filter parameters were considered unknowns.
2. A four-quarter symmetry was assumed. This means that only the parameters of one quarter of the window were considered unknowns. The rest of the window parameters were obtained through symmetry.

3. Diagonal-symmetry was assumed. In this case only half of the parameters on the diagonal of the filter window were considered unknowns. The rest of the window parameters were obtained through symmetry.
4. Using circular window with diagonal-symmetry. In this case, a circular window was assumed instead of the square window assumed in the third variation.

Using a 21x21 filter window size, the first variation, which assumes all window parameters are unknowns in the least squares matching, required solving 448 linear algebraic system at each iteration in the least squares solution. Solving a set of linear algebraic equation of this size is a slow and impractical solution. In the second variation, only one quarter of the filter window parameters were assumed unknowns in the least squares matching, the size of the linear algebraic equation that needs to be solved was reduced to 121 equations. This was considered a reasonable size compared to the size of the equations in the first variation. In the third and fourth variation, the size of the normal equation system was further reduced to 18 equations, which is extremely reasonable considering the current computer hardware capabilities.

A total of 259 small-format aerial images were matched with the IRS image. The difference in the matching coordinates between the last three variations has never exceeded 0.1 pixels and was generally less than 0.03 pixels in all the images achieving a correlation coefficient of 0.60. On the other hand, both the four-quarter symmetry and the circular symmetry variations had more computational overhead compared to the diagonal-symmetry variation.

Table A.1 in Appendix A lists the image space coordinates for both correlation and modified least squares matching for the upper left corner of the aerial images satisfying a correlation coefficient threshold of 0.60 or more, using the diagonal-symmetry variation for the modified matching technique. Table A.1 also lists the coordinates of the image center points resulting from the least squares matching process.

Final comparison between the matching accuracy of modified and traditional techniques was performed by comparing the georeferencing results of these two techniques. In the meantime, in order to discover gross matching errors, the aerial images were visually compared with the IRS image. The results of visual and automatic matching were close to each other within 1 to 2 pixels, which is approximately the limit for human matching capabilities with 5 meter resolution IRS image considering the lack of distinguished features identifying the aerial images center points.

The procedure used to visually find the center point of the aerial image on the IRS image was to identify the aerial image center point as the intersection of its two diagonals. Then both the IRS and the aerial images were compared and a point representing the center point of the aerial image was plotted on the corresponding location on the IRS image. No previous knowledge of the result of the automatic matching locations was used in this procedure. Figures 5.1 and 5.2 show two sample aerial images and their visual and automatic matching locations located on the IRS image.

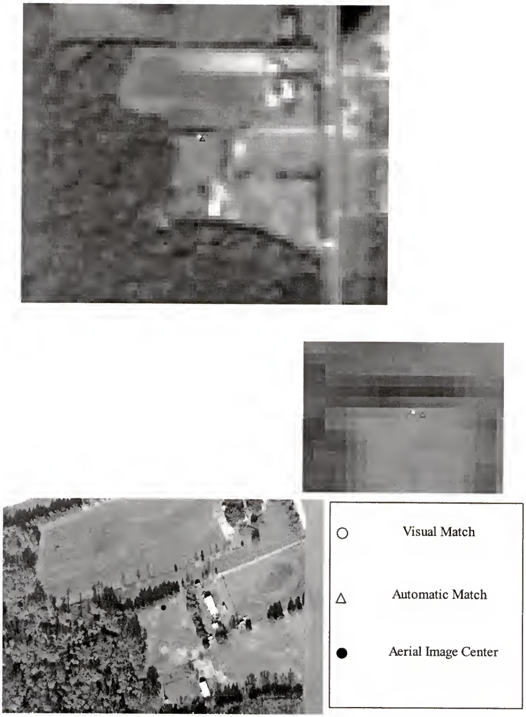


Figure 5.1. Aerial image 1227-003 center location, automatic and visual matching locations (top) plotted over the IRS image, a zoom in view for these locations (middle), and the aerial image 1227-003 with its center location marked (bottom).

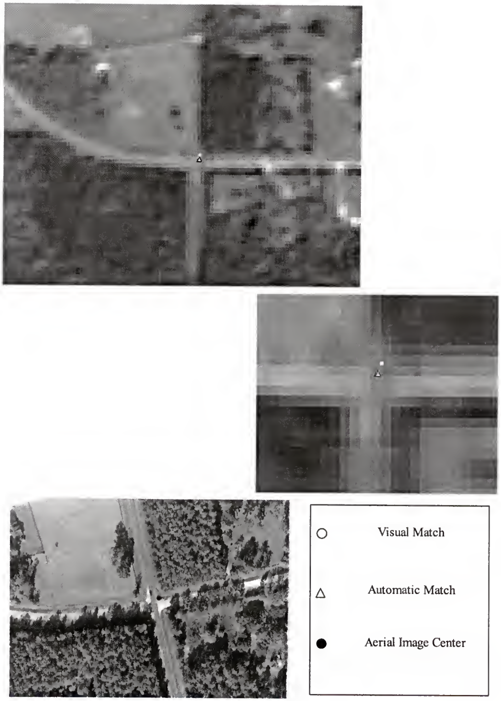


Figure 5.2. Aerial image 1227-084 center location, automatic and visual matching locations (top) plotted over the IRS image, a zoom in view for these locations (middle), and the aerial image 1227-084 with its center location marked (bottom).

Singular value decomposition analysis was performed on the linear algebraic equations of the least squares solution for some of the aerial images. The analysis indicated that the reciprocal of the condition number, the ratio between the largest and smallest elements, of the singular values diagonal matrix resulting from the analysis was on the order of 10^{-8}. This ratio is reasonably far from the critical value for double precession machine computations of 10^{-15}.

All three variations for the modified matching technique, i.e. the four-quarter, the diagonal-symmetry, the diagonal-symmetry with circular window approaches, showed close results. The diagonal-symmetry variation has an edge over the other two approaches, especially over the four-quarter approach in terms of its low computation cost. Therefore, the analysis performed through this research to compare the modified matching technique with traditional techniques and to georeference the IRS image was done using diagonal-symmetry variation.

Testing Traditional Matching Techniques

The small-format aerial images were filtered using a Gaussian filter. The images were then resampled to approximately the same resolution as the IRS image. Correlation matching was performed to provide initial values for least squares matching technique. Different Gaussian filters were tested by changing the value of the sigma parameter in the Gaussian filter. Gaussian filters with sigma values of 5,7,9, and 11 were tested. As indicated earlier, changing the sigma value changes the amount of high-frequency component filtered out of the image. The larger the sigma value the smoother the output image.

Figures 5.3 and 5.4 show small-format aerial images filtered using a Gaussian filter and resampled using the cubic convolution resampling technique. The image size was reduced from the original size by a factor of 20 in each of the x and y directions. In these figures, a blocky appearance could be distinguished in images filtered with the Gaussian filter when using a sigma value of 5. This appearance was gradually reduced when using larger sigma value. The reduction of the blocky appearance were accompanied with the increase in image smoothness as more high frequency components were eliminated from the aerial images when using larger sigma values. This was more obvious in images over urban and suburban areas where high frequency components in the images are more frequent.

Tables A.2 and A.3 in Appendix A list the results of the least squares matching for images satisfying correlation coefficient larger than 0.60 using different sigma values for the Gaussian filter. Visual checks have been conducted on the matching results of these images. Only about 4 images showed a difference of more than 2 pixels between visual and automatic matching locations. These results were adequate within the human capabilities to match 5 meter resolution images with no prominent features at the center of the aerial images.

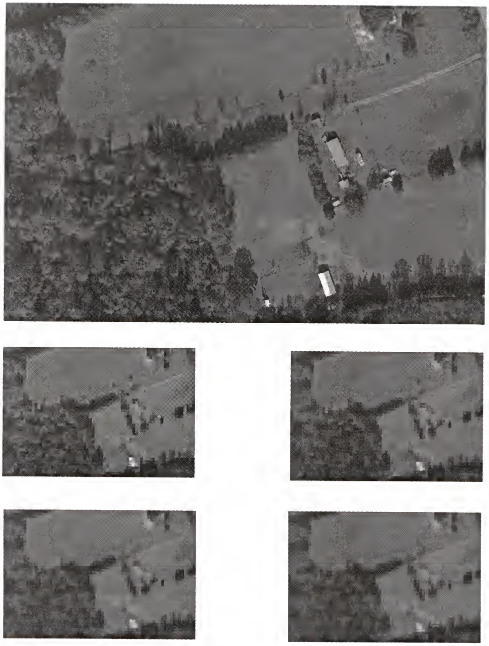
Georeferencing Satellite Image

An affine transformation model was assumed to describe the geometrical distortion of the IRS image. In this model, only 3 points are needed to solve for the six unknown transformation parameters. The matching process provides a point for each matched image and, a redundant number of images (i.e. more than 3) can be used to solve for the unknown affine transformation parameters. Therefore, a least squares solution that utilized redundant number of matched aerial images was used to determine best estimate for the affine unknown parameters (Wolf and Ghilani, 1997).

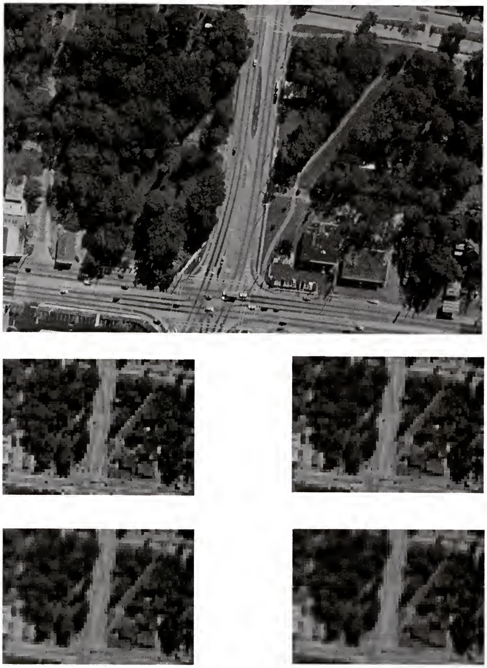
The number of control points is considered one of the main factors affecting the accuracy of the process of georeferencing satellite images. This is critical in the proposed method due to the characteristics of the matched aerial and satellite images and due to the fact that the control points used in the georeferencing process were acquired through an automatic matching process. Accordingly, different numbers of aerial images were used to georeference the IRS image. The results of the correlation matching step were used to select the matched images included in the georeferencing process.

Three different correlation coefficient thresholds were used. Images that provided a correlation coefficient that exceeded 0.70, 0.65, and 0.60 were tested. In other words, images that satisfied or exceeded the pre-specified correlation coefficient were used to georeference the IRS image. The row and column numbers resulting from the least squares matching together with the corresponding ground coordinates were used to determine the affine transformation parameters describing the satellite image geometrical distortions. Both traditional and modified least squares matching results were used.

Two sets of ground coordinates of the center point of the matched images were used. The first set was obtained through the one-meter resolution DOQQ. The other set of ground coordinates was obtained through the onboard GPS and attitude data simultaneously acquired with the small-format aerial images. The methods and computations used to obtain these coordinates were described in Chapter 3.



Figures 5.3. Small-format aerial image 1227-003 (top image) filtered with different Gaussian filters with different sigma values and resampled using cubic convolution resampling technique. Used sigma values are 5 (middle left image), 7(middle right), 9 (lower left), and 11 (lower right).



Figures 5.4. Small-format aerial image 1227-051 (top image) filtered with different Gaussian filters with different sigma values and resampled using cubic convolution resampling technique. Used sigma values are 5 (middle left image), 7 (middle right), 9 (lower left), and 11 (lower right).

Once the unknown affine transformation parameters were obtained, 30 checkpoints distributed over the IRS image were used to test the overall accuracy of the georeferencing technique. The row and column coordinates of these checkpoints were identified on the IRS image. The difference between calculated and known ground coordinates of the checkpoints in both the x and y directions were computed. The root mean square error (RMSE) in each of the x and y directions was calculated. These results were computed using the affine transformation parameters obtained from images matched using both the traditional and modified least squares matching techniques.

The results from the traditional matching method with different Gaussian filters were performed. These results were performed for images exceeding different correlation thresholds of 0.60, 0.65, and 0.70. Tables B.1 to B.4 in Appendix B show the differences in the x and y directions for the checkpoints together with the obtained RMS values when using Gaussian filter with sigma equals to 5, 7, 9, and 11. Table B.5 in Appendix B shows these differences when using the modified least squares matching technique.

The ground coordinates obtained using the DOQQ images are believed to be more accurate than those obtained through the onboard navigation sensors (Abd-Elrahman et al., 2000). Although, the developed georeferencing technique depends on using the navigation sensors data, the DOQQ ground coordinates are more accurate and useful to compare the accuracy of different matching techniques. The comparison between the accuracy of the traditional and the modified image matching techniques will be discussed in detail in Chapter 6. Generally, a sub pixel accuracy of georeferencing the IRS satellite image was achieved in this case.

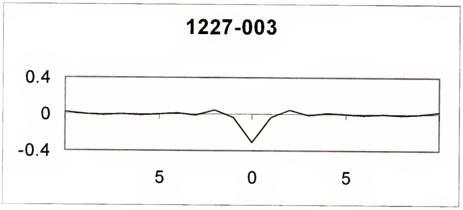
Finally, the affine transformation parameters obtained using the modified matching technique, where the ground coordinates of the matched image center points are computed from the navigation sensors data, were computed and used for georeferencing the IRS satellite image. Table B.6 in Appendix B lists the differences between the computed and known ground coordinates of the checkpoints in this case. Due to the accuracy of the navigation sensor data, the computed RMS for the difference in both the x and y directions were less than 2 pixels in almost all used correlation thresholds.

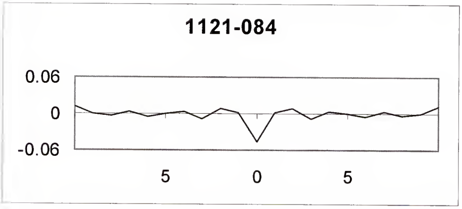
Image Filter Parameters

The output filter parameters resulting from the least squares matching process indicated two prominent patterns. Figures 5.5 and 5.6 show these filter parameters for some aerial images. Although, the output filter values change from one image to the other, the general shape of the filters for most of the images satisfying a correlation threshold of 0.60 were similar to those shown in these figures. Comparing the shape of this filter to those represented in Chapter 2 indicated a band-pass filter. This argument is supported by the filters frequency domain analysis presented in Chapter 6.

Visual Comparison of Filtered Images

Small format aerial images resulting from the modified least squares matching technique were visually compared with corresponding areas in the IRS image. These images were filtered and resampled to match the IRS resolution within the modified matching technique so that one pixel in the output resampled image was extracted from nxn window of pixels in the original image and corresponds to one pixel in the IRS image as shown in Figure 4.2. Figures 5.7, 5.8, and 5.9 show sample images filtered using the filter parameter resulting from the new matching technique. The figures show visual similarity between the IRS satellite image and the small-format aerial image filtered with the output filter parameters.





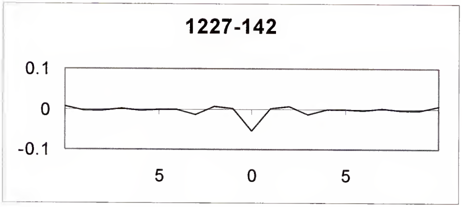
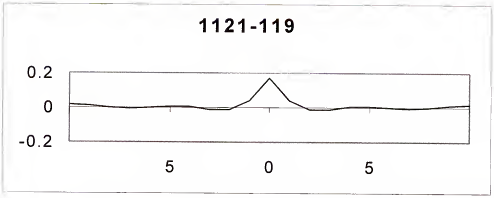
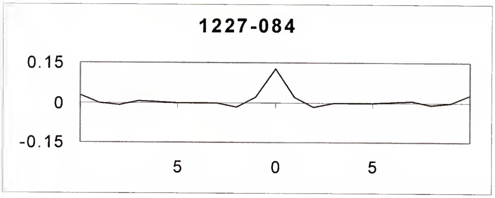


Figure 5.5. First pattern of the output filter parameters





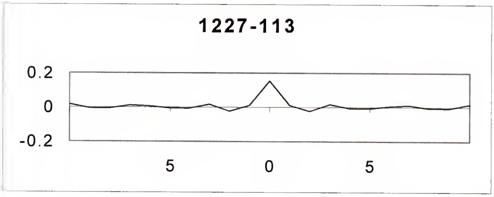
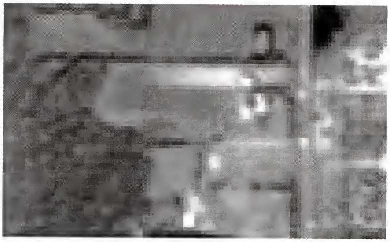
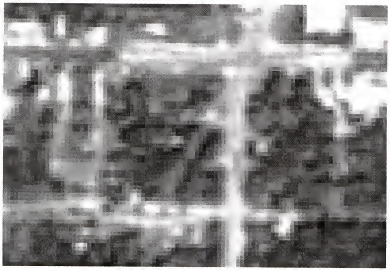
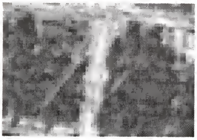


Figure 5.6. Second pattern of the output filter parameters

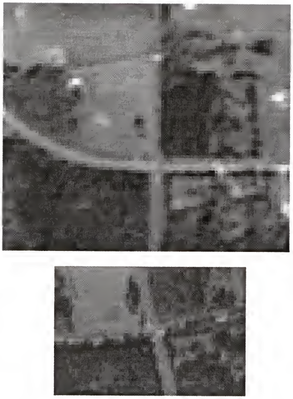


Figures 5.7. Small-format aerial image 1227-003 filtered using the filter parameter and resampled within the new matching technique (bottom) and surrounding area from the IRS satellite image (top).





Figures 5.8. Small-format aerial image 1227-051 filtered using the filter parameter and resampled within the new matching technique (bottom) and surrounding area from the IRS satellite image (top).



Figures 5.9. Small-format aerial image 1227-084 filtered using the filter parameter and resampled within the new matching technique (bottom) and surrounding area from the IRS satellite image (top).

CHAPTER 6
EVALUATION AND DISCUSSION

In this chapter, the results of georeferencing the satellite images presented in Chapter 5 are evaluated. The accuracy of traditional and modified least squares matching techniques are compared. The output filter parameters resulting from the modified least squares matching techniques are discussed.

Satellite Image Georeferencing Evaluation

Different statistics were calculated for the differences between computed and known values for the checkpoint coordinates in both the x and y directions. These statistics include mean, standard deviation and RMSE. Tables 6.1, 6.2, and 6-3 summarize these values using correlation coefficient threshold of 0.60, 0.65, and 0.70 respectively.

Table 6.1. RMSE, mean, standard deviation for the differences between known and computed coordinates of 30 check points, correlation threshold equal to 0.60.

Matching Method		Correlation > 0.60					
		dx statistics (meter)			dy statistics (meter)		
		RMSE	Mean	stdev	RMSE	Mean	stdev
Modified LS Matching		3.77	-1.03	3.69	2.51	-1.01	2.34
Traditional Matching	sigma=5	4.34	-1.45	4.16	3.75	-1.89	3.29
	sigma=7	4.10	-1.23	3.98	3.70	-1.73	3.32
	sigma=9	4.12	-1.29	3.98	3.64	-1.70	3.27
	sigma=11	4.20	-1.42	4.02	3.65	-1.73	3.27

Table 6.2. RMSE, mean, and standard deviation for the differences between known and computed coordinates of 30 check points, correlation threshold equal to 0.65.

Matching Method		Correlation > 0.65					
		dx statistics (meter)			dy statistics (meter)		
		RMSE	Mean	stdev	RMSE	Mean	stdev
Modified LS Matching		4.20	-1.37	4.04	3.00	-1.37	2.71
Traditional Matching	sigma=5	4.85	-1.85	4.75	4.20	-2.29	3.52
	sigma=7	4.47	-1.85	4.14	4.14	-1.92	3.73
	sigma=9	4.56	-1.98	4.17	4.08	-1.89	3.68
	sigma=11	4.65	-2.12	4.21	4.05	-1.88	3.64

Table 6.3. RMSE, mean, and standard deviation for the differences between known and computed coordinates of 30 check points, correlation threshold equal to 0.70.

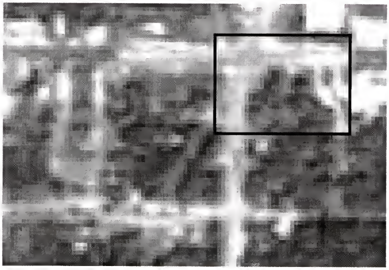
Matching Method		Correlation > 0.70					
		dx statistics (meter)			dy statistics (meter)		
		RMSE	Mean	stdev	RMSE	Mean	stdev
Modified LS Matching		4.31	-2.09	3.83	4.22	-2.75	3.26
Traditional Matching	sigma=5	5.04	-2.50	4.62	5.67	-3.74	4.27
	sigma=7	5.17	-3.07	4.23	5.71	-3.41	4.66
	sigma=9	5.28	-3.18	4.28	5.58	-3.30	4.58
	sigma=11	5.38	-3.29	4.34	5.51	-3.23	4.54

The results presented in the previous tables show that the obtained RMSE values were less than 5 meters (one pixel) using the modified matching technique. The same results were obtained with all tested correlation coefficient thresholds. RMSE computed for the traditional matching technique results was less than 5 meters in all tested variations of Gaussian filter using correlation coefficient of 0.60 and 0.65. However, the RMSE was slightly more than 5 meter in case of using 0.70 correlation coefficient threshold.

These RMSE results indicate that both traditional and modified matching techniques were able to provide a matching accuracy that achieved sub-pixel georeferencing accuracy even though the ground coordinates of the images' center points were extracted from the USGS DOQQ images. In the limited cases where the RMSE was above the pixel level, the traditional matching technique was used with correlation coefficient threshold of 0.70. This could be attributed to the lower accuracy of the traditional matching technique compared to the modified one and to the less number of images used in the georeferencing process in case of using high correlation coefficient threshold.

Generally, The RMSE values in the x direction were found to be larger than the corresponding ones in the y direction. These differences did not affect the overall results of obtaining sub-pixel root mean square error for the georeferencing process. Many reasons can contribute and cause these differences. Knowing that the aerial images and the satellite image were captured at a different date and time increases the possibility that both shadow and sun angle might have affected the accuracy of the image matching process.

The IRS image was captured before noon while the aerial images were taken in the late afternoon. This was clearly shown as the shadow extends to the west in the satellite image while it extends to the east in the aerial images. On the other hand, the shadows in both images extend to the north. Figure 6.1 shows the shadow extension in both an aerial image and the IRS image. The effects of the shadow may vary from one image to the other depending on the type of the image features, their locations in the image, sun angle, and feature heights. Having a shadow that extends in two different directions, i.e. to the east in the aerial image and to the west in the IRS image, increases the chance of having larger standard deviation values for the x direction results compared to the results in the y direction. In other words, the difference in shadow direction between matched images might have contributed to the lower precision of the matching process in the x direction while it has less effect on the obtained matching results in the y direction. More investigation is recommended to thoroughly study the effect of both shadow and sun angle directions on the automatic matching process.



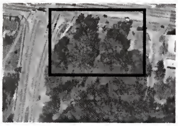


Figure 6.1. Part of IRS image(top) showing the trees shadow extends in the north west direction and part of aerial image 1227-051(bottom) showing the shadow extending in the north east direction.

The t-student statistical test was conducted on the mean of the discrepancies between the computed and known coordinates in both the x and y directions. The critical t-student statistic at certain confidence level was determined from the statistical tables using a number of degrees of freedom calculated as follows:

degrees of freedom = # of checkpoints - 1 = 29

Table 6.4 lists the computed *t*-statistic values computed for each of the x and y discrepancies for the test results. Only the results obtained using correlation thresholds of 0.60 and 0.65 were listed. The results using correlation threshold of 0.70 was less important as the number of used images was small, about 7 images, and the resulting RMSEs were much larger than those obtained using correlation thresholds of 0.60 and 0.65. The *t* statistic was computed using the mean and the standard deviation values and the equation presented in Chapter 2. The following null and alternative hypothesis were used:

$$H_0 : \bar{x} = 0$$
$$H_a : \bar{x} \neq 0$$

The computed statistic for the modified matching technique results were less than the critical value of 2.045 at the 95% confidence level in the x direction when using correlation threshold of 0.60 and 0.65. For the y direction results, the *t*-student statistic was less than the critical value of 2.415 at the 97% confidence level when using correlation threshold of 0.60. This means that, the null hypothesis, i.e. the equality between sample mean and a population mean of zero, could not be rejected at these specified confidence levels.

However, the results of the statistic computed for the traditional matching technique result was less than the critical value at the 95% confidence level only in the x direction when using 0.60 correlation threshold. In the y direction, the null hypothesis could be rejected at both 95% and 97% confidence levels for all correlation threshold values.

Table 6.4. *t*-student statistic in the x and y direction for the mean values of the discrepancies between known and computed coordinates of 30 checkpoints.

Matching Method		Correlation > 0.60		Correlation > 0.65	
		t-student statistic (x)	t-student statistic (y)	t-student statistic (x)	t-student statistic (y)
Modified LS Matching		-1.51	-2.31	-1.83	-2.72
Traditional Matching	sigma=5	-1.88	-3.09	-2.10	-3.50
	sigma=7	-1.66	-2.81	-2.41	-2.77
	sigma=9	-1.75	-2.81	-2.56	-2.78
	sigma=11	-1.90	-2.85	-2.71	-2.78

These results indicate the higher possibility of having systematic errors in the georeferencing process in case of using small number of matched images, i.e. using 0.70 correlation coefficient threshold and when using traditional matching techniques. Nevertheless, the overall computed RMSE in most if not all these cases still indicate a sub-pixel georeferencing accuracy, which undermines the effect of existing systematic errors, if any, in the matching process.

It should be mentioned here that the high value of the t-student statistic computed for the y direction results compared to those computed for the x direction results is mainly attributed to the high standard deviation value computed for the x discrepancies. These high standard deviation values can in turn be attributed to the spread in the x discrepancies and again raises the need to study the effect of the sun angle and shadow effects on the matching process accuracy, which is recommended as extension to this research.

In the following, a comparison between the accuracy of the modified and traditional matching techniques and the effect of the number of images used in the georeferencing process is discussed. In addition, the accuracy of the georeferencing process when using the navigation data to provide the ground coordinates of the aerial images center points is presented.

Modified vs. Traditional Matching Techniques

Both modified and traditional matching techniques were tested for georeferencing IRS satellite image. Figures 6.2 and 6.3 show the RMSE for the differences between computed and known checkpoint coordinates in case of using the traditional and modified matching techniques in the x and y directions, respectively. The figures show that the RMSE values obtained from the results of the traditional matching technique using images filtered with different variations of the Gaussian filter were higher than those obtained using the modified matching technique.

These results indicate that the modified matching technique, which involves solving for the image filter parameters as unknowns within the least squares matching process, provided more accurate matching results than traditional matching techniques tested with variations of Gaussian sigma values. Although the difference in RMSE computed for the discrepancies in the x direction using traditional and the modified matching techniques results was small, it is consistent using different correlation coefficient thresholds and using all tested variations of the Gaussian filter parameters.

The F-statistic test was conducted as explained in Chapter2 to compare the standard deviation or the variance of the discrepancies obtained using the modified and traditional matching techniques. Only the georeferencing results using a correlation coefficient threshold of 0.60 was tested to represent the most accurate obtained results. The F-statistic values computed using Equation (2-53) are listed in Table 6.5.

The critical F statistic was computed at the 90% and 95% confidence levels as follows:

degrees of freedom for the first and second sample = number of checkpoints - 1 = 29

$F_{critical}$ at 95% confidence level, ν_1 and ν_2 (one-tailed test) =1.86

$F_{critical}$ at 90% confidence level, ν_1 and ν_2 (one-tailed test) =1.62

The computed F statistic for the y discrepancies for the first two variations of the traditional matching technique results, i.e. when using a Gaussian filter with sigma equals to 5 and 7, was larger than the critical F statistic at the 90% confidence level. The computed F statistic for the y discrepancies for the second two variations, i.e. when using a Gaussian filter with sigma equals to 9 and 11, was larger than the critical F statistic at the 95% confidence level.

This means that the null hypothesis could be rejected for the y direction results at the 95% and 90% confidence level for the first and second two variations of the traditional matching technique results respectively. In other words, variance computed for the y discrepancies computed using any of the traditional matching technique results was significantly different than the corresponding variance computed from the modified matching technique at the 90% confidence level.

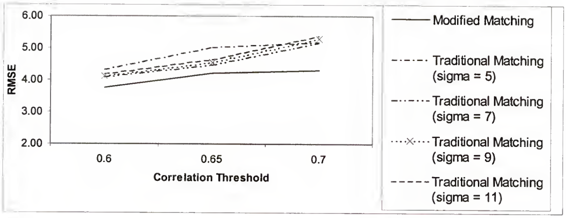


Figure 6.2. RMSE of the differences between the known and computed coordinates of checkpoints in the x direction computed using images satisfying different correlation thresholds.

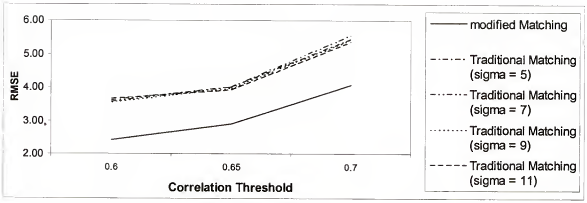


Figure 6.3. RMSE of the differences between the known and computed coordinates of checkpoints in the y direction computed using images satisfying different correlation thresholds.

Table 6.5. F statistic in the x and y direction for the discrepancies between known and computed coordinates of 30 checkpoints in case of using correlation threshold of 0.60.

		F statistic (x)	F statistic (y)
Traditional Matching	sigma=5	1.09	1.76
	sigma=7	1.09	1.76
	sigma=9	1.15	1.97
	sigma=11	1.16	1.95

On the other hand, the null hypothesis could not be rejected at the 95% or 90% confidence level for the x discrepancies. However, the standard deviation of the x discrepancies computed using the modified matching technique results were still consistently less than the standard deviation computed using traditional matching results with different variations of Gaussian filter parameters and even with different correlation coefficient thresholds.

Correlation Thresholds and needed control points

The number of control points needed for georeferencing satellite images has been the topic of many research. Traditional surveying observations or GPS observations can provide control points whose accuracy can achieve centimeter or even millimeter level. In this research, control points were provided by matching small-format aerial images with the IRS satellite image.

This emphasizes the need to investigate the effect of the number of matched images on the accuracy of the georeferencing process. The correlation coefficient was used in this research to select the images used for georeferencing the satellite image. As described previously, different correlation coefficient thresholds were used to decide on which images were used in the least squares transformation process. Increasing the value of the correlation threshold limits the number of images participating in the georeferencing process. All the results shown previously for both modified and traditional matching techniques indicated that the RMSE values decreases with the increase in the number of used control points. This means that by using more control points, the accuracy of the georeferencing process increased.

Matching a whole small-format aerial image that includes many sources of distortions such as lens distortions, shadow effect, relief displacements, …etc raises the probability of having biased matching especially when using an affine transformation model to describe these distortions. In order to overcome these sources of errors, more matched images should be included in the georeferencing process to localize existing errors in the matching process. Studying the effect of image distortions on the matching process is recommended as extension to this research.

However, further increasing the number of images participating in the georeferencing process by decreasing the value of the correlation coefficient threshold added more images with higher potential for matching errors to the solution. In this case matching errors could result in low accuracy georeferencing results. This conclusion was supported by the number of matched images that were rejected based on outlier rejection technique applied to the residuals resulting from the georeferencing process. While only one image was revoked from the georeferencing process due to large residuals that was recognized as outlier in case of using 0.65 as correlation coefficient threshold, the number of rejected images were increased to 4 images when using 0.60 correlation coefficient threshold.

Georeferencing using Navigation Data

Up to this point, the ground coordinates of the center point of the used small-format aerial images were acquired through visual interpretation from USGS DOQQ. The USGS DOQQ have one-meter ground resolution and meets the national map accuracy standards for scale 1:12,000. The accuracy of these interpreted coordinates proved to be reasonable in comparing the matching accuracy of the different investigated matching techniques.

However, one of the main objectives of this research is to study the overall accuracy of georeferencing satellite images using small-format aerial images whose center point ground coordinates are computed using navigation sensors. Accordingly, the georeferencing process was repeated using these coordinates and based on the new developed matching technique results.

GPS and airplane attitude and heading measurements were recorded for each captured small-format aerial image. These data were used to compute the ground coordinates of the aerial image center points. The accuracy of the computed coordinates is expected to be less than those provided by the USGS DOQQ.

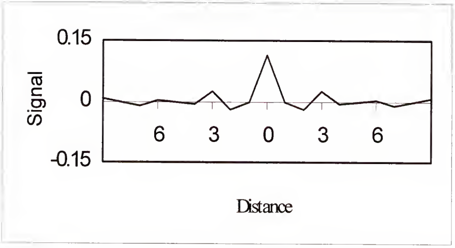
The satellite image georeferencing results, showed that a root mean square error of 6.9 and 8.3 meters in the discrepancies in the x and y directions, respectively, were achieved. The obtained RMSE in this case proved that using the navigation sensor data, less than 2 pixels RMSE could be obtained in georeferencing the IRS satellite image. Although these RMSE values were higher than the sub-pixel RMSE obtained previously, they were accepted considering the precision of the navigation sensors discussed in Chapter 3 and knowing that much more advanced integrated navigation sensors are now commercially available with reasonable prices.

Evaluating Output Filter Parameters

In the modified least squares matching technique, the filter parameters used to filter the small-format aerial images were assumed to be unknowns in the least squares matching solution. Accordingly, not only the matching parameters were obtained as an output from the matching process, but also the unknown window filter parameters. Hence these parameters relate the high-resolution aerial images to the lower resolution satellite image, generally, they were expected to represent a low pass filter parameters.

However, some other sources of image distortions could affect these parameters and change the expected relationship between the two matched images. These sources include the noise level in both images, geometrical distortions due to image orientation and/or relief displacement, different light conditions either within the same image or between matched images. Modeling each of these sources in the least squares matching process and studying its effect on the matching process is outside the scope of this research and recommended for future investigation.

Figures 6-4 to 6-5 show samples of the obtained window filter parameter in both spatial and frequency domains for some of the matched small-format aerial images using the diagonal-symmetry variation of the modified matching technique. The shape of the filter representation in both domains suggests a band-pass filter. Some small ripple effect was shown in the frequency domain representation of the filters. Again this is expected as the filter parameters are extracted through a least squares matching process without modeling all possible sources of differences between the two matched images.



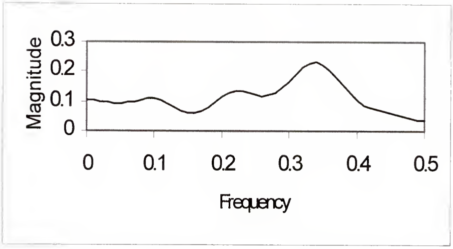
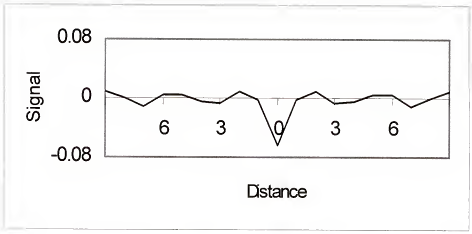


Figure 6-4. Unknown filter parameters for image 1121-095 in spatial (top) and frequency domains (bottom).



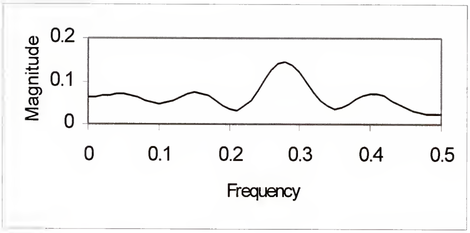
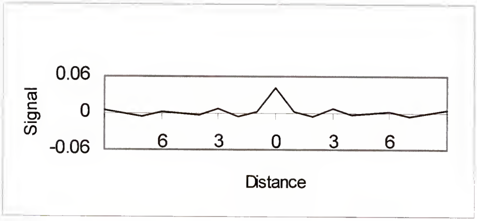


Figure 6-5. Unknown filter parameters for image 1121-074 in spatial (top) and frequency domains (bottom).



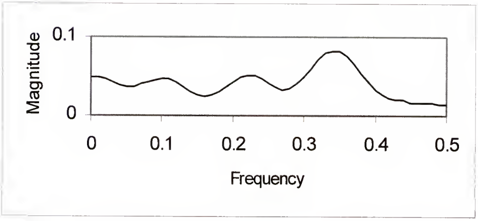


Figure 6-6. Unknown filter parameters for image 1121-071 in spatial (top) and frequency domains (bottom).

CHAPTER 7
SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

In this chapter, a summary of the developed technique and achieved results are given. The conclusions based on the discussions presented in Chapter 5 and Chapter 6 are presented. Finally, the potential future extensions of this research are introduced.

Summary

Small-format digital aerial imagery captured with a one-engine airplane equipped with a low-cost digital camera and navigation sensors were used to rectify an Indian Remote Sensing (IRS) Satellite image. This IRS image represents a class of increasingly used medium-scale satellite images. The center point location of each of the aerial imagery was determined through the navigation sensor data recorded at each image exposure. Some tests were also conducted using center point coordinates derived from the USGS one-meter resolution Digital Orthophoto Quarter Quadrangles.

Due to the approximately 20:1 resolution ratio between the two types of images, a modified image registration technique was developed to account for this scale difference. In this modified technique, two types of parameters were considered unknowns in a least squares matching process. Both the affine geometric transformation parameters between each aerial image and the satellite image and the filter parameters used to filter the frequency content of the high-resolution image to radiometrically match the frequency content of the low-resolutions image were considered unknowns in the registration process.

The results of the modified matching technique were compared with the results of traditional matching technique, which depends on filtering the high-resolution image with a low-pass filter and resampling it to approximately the same ground pixel size of the low- resolution image. The filtered and resampled aerial image was then matched with the low-resolution image. Generally, least squares image registration techniques can provide a sub-pixel matching accuracy. A correlation matching was used before the least squares matching to provide approximate values for the affine transformation parameters needed to start the least squares matching process.

The matching accuracy of the modified and traditional techniques was compared using their results in rectifying the IRS satellite image. The matching results for certain aerial images that satisfy a matching correlation coefficient threshold were included in the rectifying process. Correlation thresholds of 0.60, 0.65,and 0.70 were used. More images were included in the georeferencing process when lower value for the correlation coefficient threshold was used. Checkpoints were selected in the IRS image in order to compare the computed object space coordinates and the known coordinates for these checkpoints. The filter parameters resulting from the least squares process were evaluated in both spatial and frequency domains.

Conclusions

This research proved that small-format aerial images could be successfully used to georeference IRS satellite images. Onboard navigation sensors were used to provide object space coordinates for the small-format aerial images. Less than two pixels root mean square error was achieved in both the x and y directions when object space coordinates of the aerial images center points were derived from the navigation sensors data while a sub-pixel root mean square error was obtained when these coordinates were obtained from USGS DOQQ.

This means that, a system of a small-format aerial camera, GPS, and attitude and heading measuring device configured on an airplane proved able to provide enough control points to georeference a satellite image with no ground observations. Although this system provided a matching accuracy of less than two pixels, more advanced commercially available systems, which utilize more accurate navigation sensors, likely provide a sub-pixel georeferencing accuracy.

A new least squares matching technique which accounts for a severe scale difference between matched images was successfully developed and demonstrated to give better matching results than traditional matching techniques in which the high-resolution images are filtered with low-pass filter, resampled and matched in separate steps. In the developed technique, the filter parameters used to filter out the high-frequency components in the small-format aerial images were left as unknowns in the least squares matching solution. In this case, the unknown filter parameters can reveal the radiometric relationship between the high and low-resolution images.

Matching a whole small-format aerial image with the IRS satellite image was successful although different kinds of distortions existed in both types of images. Sources of the distortions in the aerial images include relief displacement, lens distortions, electronic distortions, and geometrical distortions due to airplane attitude. Similarly, distortions in the IRS image were expected due to optical and electronic noise, sensor perturbations, etc. In addition, differences due to sun angle, sensor angle, shadow, and time change can also affect the matching process. This result is very important for many applications. One of these applications is to match the acquired small-format aerial images with other georeferenced imagery to detect systematic errors in computed image positions and hence, in the navigation sensor data. Different image fusion applications can also benefit from these results.

The correlation coefficient resulting from initial correlation matching provided a good measure for predicting a success of the least squares matching process. The potential for achieving false matching with large number of acquired images that needs to be matched motivated the use of the correlation coefficient as a threshold to select images that participate in the georeferencing process.

The satellite georeferencing accuracy was tested using different number of matched aerial images. Different correlation coefficient thresholds were used to select these images. The results indicated that increasing the number of images led to increase in the georeferencing accuracy. On the other hand, the results showed that adding more images increases the risk of adding images with false matching. A correlation coefficient threshold of 0.60 that provided more than 50 matched images gave the best georeferencing accuracy.

Aerial images with different landscape contents were used and accurately matched. This includes images over urban, suburban, agricultural, and mixed areas. Visual comparison was used to check the accuracy of the automatic matching process. This visual comparison was difficult to accomplish in certain areas especially over forest areas. This visual inspection of the matching accuracy showed accepted matching results, mostly less than 2 pixels difference, considering the difficulties and the limitations of the visual matching process in many cases. Distinctive features such as clear cut, water bodies, and mixed forest areas provided good matching features for images representing forest areas. Matching was noticed to fail for images taken over dense forest areas.

Recommendations

In the least squares matching technique developed here, an affine geometrical transformation model was assumed to describe the relationship between the aerial and satellite images. Further research should be conducted to use other models. Moreover, sources of distortions such as lens distortions, perspective projection, etc, may also be involved in the transformation model. Special interest should be devoted to study the effect of the shadow on the matching process. Different shadow direction and size in the matched imagery can significantly degrade the quality of the matching process especially when matched images are acquired at a different date and time.

As indicated previously, using more precise navigation sensors is recommended to test for achieved georeferencing accuracy. The system used in this research was designed to achieve a positional accuracy sufficient for assisting land cover classification for Landsat satellite imagery. Better-integrated systems that employ more precise navigation instruments are commercially available with downward moving prices.

Analyzing the output filter parameters from the modified least squares matching could reveal more information about the relationship between the matched images. For example, if the output filter is a band-reject filter, further investigation needs to be conducted to identify what frequencies in the high-resolution image are rejected in the low-resolution image. This can be done by analyzing both images and the output filter in the frequency domain. Moreover, in this research a narrow point spread function for the

IRS image was assumed. Future research is suggested to adjust the developed least squares matching technique to handle wider point spread functions.

In this research the modified least squares matching technique was used to georeference a medium resolution IRS image. With the tremendous increase in commercial satellite images captured with different types of sensors, further research is needed to test the method developed in this research with different types of sensors such as the medium-resolution SPOT images and the high-resolution IKONOS images.

The method developed in this research was tested in a relatively flat terrain, which effectively reduced image distortions due to relief displacement. More research is suggested to test this method in more diverse terrain. Although, different types of landscapes were represented in the IRS image and the aerial images, the modified matching technique need also to be tested with more landscape types such as high residential areas, desert areas, coastal areas, etc.

The correlation coefficient was successfully used in this research to select images involved in the georeferencing process. More precise and faster methods to choose images with high potential for successful and accurate matching are suggested. The aerial image texture and color information are major candidates for such methods. Pre-selecting the candidate images before the matching process will help in reducing the time spent in matching images that either give low matching accuracy or even do not match at all.

APPENDIX A
MATCHING RESULTS

Table A.1. Correlation and least squares matching results for images with correlation coefficient equal to or exceeds 0.60.

Image	correlation Coefficient	x-upper-left correlation matching	y-upper-left correlation matching	x-upper-left LS matching	y-upper-left LS matching	x-center LS matching	y-center LS matching
1227-003	0.77	4646	5363	4649.32	5363.63	4672.57	5334.93
1227-004	0.74	4617	5204	4618.00	5208.43	4648.51	5183.00
1227-006	0.66	4608	4913	4610.47	4909.98	4641.18	4884.34
1227-015	0.67	4587	3570	4592.46	3569.39	4619.91	3549.81
1227-017	0.71	4593	3278	4598.05	3280.54	4630.61	3260.94
1227-020	0.62	4640	2833	4640.42	2831.23	4673.81	2809.57
1227-048	0.68	4093	1209	4093.94	1206.12	4126.60	1188.07
1227-050	0.61	4088	1512	4087.33	1506.35	4120.50	1489.84
1227-051	0.71	4102	1661	4101.41	1657.54	4133.20	1639.25
1227-056	0.65	4137	2406	4135.38	2401.56	4167.14	2385.90
1227-083	0.61	3475	5062	3479.54	5064.01	3506.54	5035.04
1227-084	0.68	3468	4905	3469.24	4907.61	3498.09	4881.10
1227-087	0.63	3465	4468	3467.09	4470.88	3497.07	4444.18
1227-098	0.65	3496	2841	3500.81	2842.74	3531.91	2818.53
1227-103	0.62	3500	2100	3500.68	2100.08	3534.02	2076.40
1227-111	0.72	3506	914	3503.30	915.36	3538.88	891.76
1227-112	0.64	3514	767	3515.43	767.61	3543.76	739.53
1227-113	0.71	3492	623	3491.84	622.66	3519.52	596.56
1227-115	0.74	3479	311	3479.96	312.99	3511.41	286.71
1227-122	0.64	2861	633	2858.11	627.72	2891.37	609.84
1227-123	0.69	2874	778	2871.77	772.98	2906.29	756.52
1227-142	0.61	2775	3583	2776.98	3580.45	2808.22	3565.44
1121-006	0.79	2407	4916	2411.02	4919.74	2446.05	4908.20
1121-007	0.63	2428	4767	2430.43	4770.38	2464.97	4755.21
1121-011	0.71	2452	4174	2459.05	4173.83	2496.56	4160.11
1121-014	0.63	2432	3728	2437.04	3731.39	2473.17	3713.06
1121-017	0.63	2429	3276	2431.50	3281.21	2469.76	3264.95
1121-035	0.70	2326	601	2328.56	606.65	2368.24	596.58
1121-037	0.65	2355	316	2356.25	318.92	2394.40	302.93

Table A.1. Continued

Image	correlation Coefficient	x-upper-left correlation matching	y-upper-left correlation matching	x-upper-left LS matching	y-upper-left LS matching	x-center LS matching	y-center LS matching
1121-038	0.68	2343	163	2347.02	169.32	2383.00	152.53
1121-052	0.74	1664	1818	1664.90	1811.81	1694.43	1792.02
1121-056	0.72	1661	2412	1660.65	2410.37	1685.88	2386.49
1121-057	0.77	1685	2560	1687.71	2558.05	1707.88	2531.96
1121-071	0.64	1755	4630	1754.29	4627.20	1785.32	4605.75
1121-072	0.75	1752	4776	1752.63	4770.62	1783.98	4750.84
1121-074	0.70	1738	5073	1736.15	5069.13	1767.63	5048.01
1121-081	0.74	1112	5363	1116.27	5366.16	1153.20	5347.69
1121-084	0.62	1113	4919	1115.17	4921.55	1149.25	4901.60
1121-095	0.72	1054	3289	1055.65	3291.58	1094.14	3273.07
1121-105	0.72	1056	1799	1059.01	1798.58	1090.87	1784.85
1121-106	0.64	1043	1650	1047.34	1653.60	1081.39	1637.20
1121-108	0.61	1042	1355	1044.02	1357.90	1082.04	1341.86
1121-116	0.65	986	173	988.15	177.85	1024.23	162.78
1121-119	0.87	495	236	493.76	235.92	520.39	210.77
1121-120	0.61	509	391	505.67	391.13	533.99	365.92
1121-125	0.63	458	1117	454.76	1114.55	487.83	1098.47
1121-127	0.62	431	1427	429.54	1426.26	457.00	1402.30
1121-128	0.77	445	1577	443.03	1576.92	471.67	1549.20
1121-130	0.64	461	2316	458.15	2311.90	488.33	2292.03
1121-131	0.79	456	2465	456.12	2461.96	484.30	2438.44

Table A.2. Least squares matching results for images with correlation coefficient equal to or exceeds 0.60 using Gaussian filter with sigma equals to 5 and 7.

	sigma = 5		sigma = 7	
Image	x-center LS matching (pix)	y-center LS matching (pix)	x-center LS matching (pix)	y-center LS matching (pix)
1227-003	4672.61	5334.96	4672.58	5334.96
1227-004	4648.50	5182.90	4648.47	5182.91
1227-006	4641.04	4884.35	4641.01	4884.35
1227-015	4620.39	3549.65	4620.36	3549.64
1227-024	4718.53	2213.54	4718.33	2213.74
1227-048	4126.72	1188.08	4126.75	1188.09
1227-050	4120.57	1489.78	4120.59	1489.75
1227-051	4133.28	1639.33	4133.29	1639.33
1227-056	4167.09	2385.97	4167.11	2385.96
1227-083	3506.30	5034.63	3506.27	5034.63
1227-084	3498.05	4881.21	3498.05	4881.24
1227-087	3497.15	4444.27	3497.12	4444.26
1227-098	3531.96	2818.62	3531.93	2818.56
1227-103	3534.22	2076.27	3534.21	2076.28
1227-111	3539.42	891.79	3539.38	891.77
1227-112	3543.81	739.76	3543.81	739.75
1227-113	3519.71	597.02	3519.63	597.03
1227-115	3511.52	286.93	3511.49	286.94
1227-122	2891.40	610.18	2891.42	610.14
1227-123	2906.41	756.71	2906.40	756.67
1227-142	2808.16	3565.57	2808.14	3565.55
1121-006	2445.94	4908.10	2445.91	4908.03
1121-007	2464.97	4755.10	2464.93	4755.09
1121-011	2495.84	4159.93	2495.80	4159.94
1121-014	2473.20	3712.99	2473.22	3712.97
1121-017	2469.87	3265.01	2469.85	3265.02
1121-035	2368.19	596.42	2368.17	596.41
1121-037	2394.34	302.75	2394.33	302.75
1121-038	2382.97	152.56	2382.95	152.55
1121-052	1694.54	1791.97	1694.56	1791.96
1121-056	1686.00	2386.45	1686.02	2386.45
1121-057	1707.89	2531.93	1707.91	2531.91
1121-071	1785.28	4605.79	1785.29	4605.78
1121-072	1783.96	4750.92	1783.96	4750.93
1121-074	1767.74	5048.04	1767.77	5048.04
1121-076	1752.69	5345.96	1752.42	5345.89
1121-081	1153.21	5347.64	1153.19	5347.63
1121-084	1149.29	4901.68	1149.26	4901.68
1121-095	1094.25	3272.63	1094.22	3272.63
1121-105	1090.68	1786.17	1090.67	1786.18

Table A.2. Continued.

Image	x-center LS matching (pix)	y-center LS matching (pix)	x-center LS matching (pix)	y-center LS matching (pix)
1121-106	1081.40	1637.13	1081.37	1637.13
1121-108	1082.15	1341.90	1082.12	1341.89
1121-111	1058.60	903.73	1058.63	903.72
1121-116	1024.19	162.74	1024.17	162.76
1121-119	520.40	210.76	520.43	210.74
1121-120	534.06	365.92	534.09	365.90
1121-125	487.90	1098.50	487.93	1098.53
1121-127	457.02	1402.38	457.03	1402.38
1121-128	472.37	1549.34	471.55	1550.45
1121-130	488.37	2291.75	488.40	2291.76
1121-131	484.39	2438.34	484.42	2438.33

Table A.3. Least squares matching results for images with correlation coefficient equal to or exceeds 0.60 using Gaussian filter with sigma equals to 7 and 9.

	sigma = 9		sigma = 11	
Image	x-center LS matching (pix)	y-center LS matching (pix)	x-center LS matching (pix)	y-center LS matching (pix)
1227-003	4672.56	5334.97	4672.55	5334.97
1227-004	4648.45	5182.90	4648.43	5182.90
1227-006	4640.97	4884.34	4640.94	4884.34
1227-015	4620.33	3549.64	4620.29	3549.66
1227-024	4718.17	2214.07	4718.02	2214.25
1227-048	4126.78	1188.09	4126.81	1188.08
1227-050	4120.61	1489.73	4120.62	1489.71
1227-051	4133.31	1639.32	4133.31	1639.32
1227-056	4167.13	2385.96	4167.15	2385.96
1227-083	3506.23	5034.64	3506.23	5034.63
1227-084	3498.04	4881.25	3498.02	4881.27
1227-087	3497.11	4444.28	3497.10	4444.30
1227-098	3531.91	2818.55	3531.88	2818.55
1227-103	3534.18	2076.28	3534.16	2076.29
1227-111	3539.37	891.74	3539.34	891.72
1227-112	3543.81	739.75	3543.79	739.75
1227-113	3519.56	597.03	3519.50	597.03
1227-115	3511.47	286.95	3511.46	286.97
1227-122	2891.42	610.16	2891.42	610.18
1227-123	2906.42	756.64	2906.43	756.63
1227-142	2808.14	3565.53	2808.12	3565.51
1121-006	2445.89	4908.00	2445.87	4907.97
1121-007	2464.90	4755.06	2464.87	4755.05
1121-011	2495.77	4159.95	2495.72	4159.96
1121-014	2473.23	3712.95	2473.23	3712.91
1121-017	2469.82	3265.01	2469.81	3265.00
1121-035	2368.11	596.40	2368.09	596.37
1121-037	2394.31	302.76	2394.30	302.77
1121-038	2382.92	152.54	2382.90	152.54
1121-052	1694.58	1791.95	1694.60	1791.95
1121-056	1686.04	2386.44	1686.06	2386.43
1121-057	1707.94	2531.90	1707.96	2531.88
1121-071	1785.31	4605.77	1785.33	4605.76
1121-072	1783.97	4750.93	1783.97	4750.94
1121-074	1767.79	5048.04	1767.82	5048.04
1121-076	1752.08	5345.71	1751.71	5345.53
1121-081	1153.17	5347.63	1153.13	5347.63
1121-084	1149.24	4901.67	1149.21	4901.66
1121-095	1094.21	3272.59	1094.19	3272.51
1121-105	1090.65	1786.18	1090.64	1786.17

Table A.3. Continued

	sigma = 9		sigma = 11	
Image	x-center LS matching (pix)	y-center LS matching (pix)	x-center LS matching (pix)	y-center LS matching (pix)
1121-106	1081.34	1637.12	1081.31	1637.12
1121-108	1082.10	1341.87	1082.08	1341.85
1121-111	1058.62	903.72	1058.59	903.74
1121-116	1024.15	162.76	1024.13	162.76
1121-119	520.46	210.73	520.49	210.72
1121-120	534.11	365.90	534.13	365.91
1121-125	487.98	1098.51	487.99	1098.48
1121-127	457.05	1402.38	457.07	1402.38
1121-128	471.53	1550.57	471.52	1550.71
1121-130	488.42	2291.76	488.45	2291.75
1121-131	484.44	2438.33	484.46	2438.32

APPENDIX B
GEOREFERENCING RESULTS

Table B.1. Differences between known and computed coordinates of 30 checkpoints. When using traditional matching technique with images filtered by a Gaussian filter with sigma value equals to 5.

Check Point	Correlation > 0.60		Correlation > 0.65		Correlation > 0.70	
	dx	dy	dx	dy	dx	dy
GPSchk05	-3.70	1.75	-3.48	1.97	-5.07	1.44
GPSchk06	2.88	-3.12	2.84	-2.98	1.58	-3.54
GPSchk07	0.39	2.24	0.27	2.62	-0.54	2.90
GPSchk08	-6.66	-3.46	-7.29	-4.31	-8.75	-7.47
GPSchk11	9.17	-4.81	8.84	-5.33	7.27	-7.70
GPSchk18	2.29	-1.39	1.43	-1.86	0.82	-3.63
GPSchk21	-3.68	-3.30	-4.10	-3.75	-5.43	-5.84
DOQchk01	-3.41	0.10	-3.33	0.35	-4.65	0.05
DOQchk02	-2.19	6.51	-1.88	6.74	-3.61	6.17
DOQchk03	1.27	-2.11	0.54	-2.94	-0.73	-5.94
DOQchk04	-3.73	-5.37	-4.52	-6.17	-5.66	-9.08
DOQchk05	-7.39	-2.54	-8.22	-3.34	-9.29	-6.19
DOQchk06	-0.51	-1.53	-1.44	-1.83	-1.71	-3.01
DOQchk07	-0.19	0.04	-0.45	-0.07	-1.64	-1.22
DOQchk08	-0.18	0.82	-0.47	1.09	-1.13	1.19
DOQchk09	-6.92	-5.00	-8.17	-5.98	-8.75	-9.03
DOQchk10	-1.43	1.43	-1.21	1.36	-3.16	-0.10
DOQchk11	3.69	2.15	3.63	2.06	2.12	0.78
DOQchk12	-2.14	2.54	-2.81	2.45	-3.27	1.73
DOQchk13	-0.10	-5.05	-0.65	-5.44	-1.68	-7.24
DOQchk14	-5.30	-2.38	-6.54	-3.06	-6.75	-5.13
DOQchk15	2.19	-4.29	1.82	-4.33	0.93	-5.13
DOQchk16	-5.40	-3.66	-6.58	-3.81	-6.24	-4.25
DOQchk17	-3.23	-3.34	-4.83	-3.98	-4.38	-5.63
DOQchk18	6.98	0.32	6.30	0.45	6.13	0.45
DOQchk19	-2.53	-4.39	-4.16	-5.34	-4.04	-7.95
DOQchk20	-4.69	-4.19	-5.95	-4.54	-5.72	-5.54
DOQchk21	-3.90	-6.35	-5.89	-7.25	-5.10	-9.39
DOQchk22	-8.16	-8.99	-10.00	-9.91	-9.49	-12.23
DOQchk23	1.77	0.71	0.82	0.79	1.07	0.89
RMSE	4.32	3.75	4.85	4.20	5.04	5.67

Table B.2. Differences between known and computed coordinates of 30 checkpoints. When using traditional matching technique with images filtered by a Gaussian filter with sigma value equals to 7.

Check Point	Correlation > 0.60		Correlation > 0.65		Correlation > 0.70	
	dx	dy	dx	dy	dx	dy
GPSchk05	-3.93	1.96	-3.91	2.45	-5.17	2.16
GPSchk06	2.76	-2.92	2.58	-2.55	1.48	-2.94
GPSchk07	0.20	2.45	-0.07	3.07	-0.58	3.51
GPSchk08	-6.20	-3.34	-6.71	-4.07	-9.05	-7.29
GPSchk11	9.40	-4.67	9.09	-5.01	7.03	-7.33
GPSchk18	2.66	-1.25	1.93	-1.61	0.62	-3.48
GPSchk21	-3.44	-3.15	-3.84	-3.44	-5.65	-5.51
DOQchk01	-3.62	0.31	-3.71	0.82	-4.73	0.73
DOQchk02	-2.46	6.71	-2.37	7.23	-3.72	6.94
DOQchk03	1.75	-1.99	1.16	-2.71	-1.02	-5.80
DOQchk04	-3.23	-5.24	-3.87	-5.96	-5.94	-8.96
DOQchk05	-6.89	-2.42	-7.56	-3.12	-9.57	-6.08
DOQchk06	-0.17	-1.37	-0.99	-1.57	-1.86	-2.86
DOQchk07	-0.15	0.21	-0.46	0.30	-1.79	-0.75
DOQchk08	-0.28	1.02	-0.67	1.51	-1.18	1.71
DOQchk09	-6.22	-4.89	-7.19	-5.84	-9.04	-9.14
DOQchk10	-1.54	1.62	-1.49	1.80	-3.34	0.57
DOQchk11	3.66	2.33	3.51	2.46	1.96	1.34
DOQchk12	-1.97	2.71	-2.61	2.77	-3.38	2.02
DOQchk13	0.16	-4.90	-0.35	-5.15	-1.88	-6.95
DOQchk14	-4.72	-2.25	-5.73	-2.88	-6.96	-5.18
DOQchk15	2.24	-4.12	1.83	-3.96	0.81	-4.69
DOQchk16	-5.05	-3.50	-6.09	-3.56	-6.33	-4.18
DOQchk17	-2.55	-3.22	-3.85	-3.85	-4.56	-5.82
DOQchk18	7.06	0.50	6.38	0.80	6.07	0.79
DOQchk19	-1.71	-4.29	-3.00	-5.25	-4.30	-8.21
DOQchk20	-4.24	-4.05	-5.31	-4.32	-5.85	-5.53
DOQchk21	-2.99	-6.25	-4.57	-7.21	-5.32	-9.80
DOQchk22	-7.30	-8.89	-8.75	-9.84	-9.73	-12.57
DOQchk23	1.96	0.89	1.07	1.10	1.02	1.10
RMSE	4.10	3.70	4.47	4.14	5.17	5.71

Table B.3. Differences between known and computed coordinates of 30 checkpoints. When using traditional matching technique with images filtered by a Gaussian filter with sigma value equals to 9.

Check Point	Correlation > 0.60		Correlation > 0.65		Correlation > 0.70	
	dx	dy	dx	dy	dx	dy
GPSchk05	-3.93	1.92	-3.93	2.41	-5.15	2.19
GPSchk06	2.73	-2.96	2.53	-2.59	1.48	-2.92
GPSchk07	0.15	2.35	-0.15	2.94	-0.52	3.44
GPSchk08	-6.26	-3.20	-6.80	-3.86	-9.29	-6.99
GPSchk11	9.36	-4.58	9.03	-4.87	6.87	-7.11
GPSchk18	2.55	-1.22	1.78	-1.56	0.47	-3.34
GPSchk21	-3.50	-3.08	-3.92	-3.33	-5.79	-5.32
DOQchk01	-3.63	0.25	-3.76	0.76	-4.70	0.72
DOQchk02	-2.44	6.69	-2.37	7.21	-3.69	6.97
DOQchk03	1.67	-1.86	1.06	-2.52	-1.26	-5.52
DOQchk04	-3.32	-5.13	-3.99	-5.78	-6.18	-8.69
DOQchk05	-6.98	-2.31	-7.68	-2.95	-9.80	-5.82
DOQchk06	-0.30	-1.39	-1.16	-1.59	-1.97	-2.78
DOQchk07	-0.20	0.21	-0.54	0.32	-1.85	-0.66
DOQchk08	-0.34	0.93	-0.77	1.39	-1.15	1.65
DOQchk09	-6.36	-4.78	-7.37	-5.67	-9.32	-8.86
DOQchk10	-1.53	1.66	-1.49	1.87	-3.39	0.70
DOQchk11	3.64	2.34	3.46	2.49	1.91	1.44
DOQchk12	-2.07	2.67	-2.75	2.72	-3.44	2.05
DOQchk13	0.08	-4.86	-0.46	-5.08	-2.01	-6.79
DOQchk14	-4.87	-2.22	-5.93	-2.81	-7.16	-5.01
DOQchk15	2.18	-4.15	1.73	-3.99	0.77	-4.65
DOQchk16	-5.22	-3.58	-6.31	-3.66	-6.40	-4.19
DOQchk17	-2.75	-3.22	-4.10	-3.84	-4.76	-5.71
DOQchk18	6.94	0.40	6.23	0.67	6.07	0.74
DOQchk19	-1.91	-4.22	-3.24	-5.15	-4.57	-7.99
DOQchk20	-4.41	-4.09	-5.53	-4.37	-5.97	-5.48
DOQchk21	-3.24	-6.23	-4.87	-7.17	-5.58	-9.63
DOQchk22	-7.52	-8.85	-9.03	-9.78	-9.99	-12.38
DOQchk23	1.81	0.77	0.87	0.95	1.01	1.03
RMSE	4.15	3.66	4.56	4.08	5.28	5.58

Table B.4. Differences between known and computed coordinates of 30 checkpoints. When using traditional matching technique with images filtered by a Gaussian filter with sigma value equals to 11.

Check Point	Correlation > 0.60		Correlation > 0.65		Correlation > 0.70	
	dx	dy	dx	dy	dx	dy
GPSchk05	-3.94	1.90	-3.95	2.39	-5.14	2.23
GPSchk06	2.69	-2.99	2.47	-2.62	1.48	-2.89
GPSchk07	0.09	2.26	-0.25	2.84	-0.47	3.41
GPSchk08	-6.32	-3.09	-6.87	-3.71	-9.52	-6.80
GPSchk11	9.32	-4.51	8.98	-4.77	6.72	-6.97
GPSchk18	2.44	-1.21	1.63	-1.53	0.33	-3.26
GPSchk21	-3.55	-3.04	-3.99	-3.26	-5.93	-5.20
DOQchk01	-3.67	0.21	-3.80	0.71	-4.68	0.74
DOQchk02	-2.44	6.67	-2.37	7.20	-3.67	7.01
DOQchk03	1.60	-1.77	0.97	-2.39	-1.48	-5.35
DOQchk04	-3.40	-5.05	-4.09	-5.66	-6.40	-8.53
DOQchk05	-7.07	-2.23	-7.79	-2.84	-10.02	-5.66
DOQchk06	-0.43	-1.42	-1.34	-1.61	-2.08	-2.75
DOQchk07	-0.25	0.21	-0.61	0.33	-1.91	-0.60
DOQchk08	-0.43	0.85	-0.89	1.30	-1.12	1.63
DOQchk09	-6.49	-4.71	-7.54	-5.56	-9.58	-8.70
DOQchk10	-1.52	1.69	-1.48	1.92	-3.44	0.80
DOQchk11	3.61	2.36	3.42	2.52	1.85	1.52
DOQchk12	-2.18	2.62	-2.91	2.68	-3.50	2.07
DOQchk13	0.01	-4.84	-0.56	-5.03	-2.14	-6.70
DOQchk14	-5.02	-2.20	-6.13	-2.78	-7.36	-4.92
DOQchk15	2.11	-4.18	1.63	-4.01	0.72	-4.62
DOQchk16	-5.39	-3.67	-6.55	-3.75	-6.47	-4.21
DOQchk17	-2.94	-3.25	-4.37	-3.85	-4.94	-5.66
DOQchk18	6.82	0.31	6.05	0.57	6.06	0.71
DOQchk19	-2.09	-4.19	-3.49	-5.08	-4.82	-7.88
DOQchk20	-4.58	-4.15	-5.77	-4.42	-6.09	-5.46
DOQchk21	-3.47	-6.24	-5.19	-7.16	-5.83	-9.57
DOQchk22	-7.73	-8.85	-9.32	-9.75	-10.24	-12.30
DOQchk23	1.66	0.66	0.65	0.83	0.98	0.98
RMSE	4.20	3.65	4.65	4.05	5.38	5.51

Table B.5. Differences between known and computed coordinates of 30 checkpoints when using modified matching technique.

Check Point	Correlation > 0.60		Correlation > 0.65		Correlation > 0.70	
	dx	dy	dx	dy	dx	dy
GPSchk05	-5.33	0.59	-3.71	1.14	-5.53	0.38
GPSchk06	1.65	-3.91	2.76	-3.51	1.36	-4.21
GPSchk07	-1.44	0.78	-0.28	1.54	-1.13	1.74
GPSchk08	-4.80	-1.07	-5.45	-2.22	-7.10	-5.40
GPSchk11	9.90	-3.56	10.03	-4.17	8.24	-6.65
GPSchk18	3.27	0.01	2.53	-0.60	1.94	-2.26
GPSchk21	-3.05	-2.17	-3.03	-2.71	-4.52	-4.85
DOQchk01	-5.04	-1.09	-3.64	-0.50	-5.11	-0.98
DOQchk02	-3.92	5.25	-2.14	5.84	-4.14	5.00
DOQchk03	3.12	0.26	2.34	-0.87	0.92	-3.86
DOQchk04	-1.87	-3.00	-2.76	-4.12	-4.01	-6.97
DOQchk05	-5.54	-0.19	-6.47	-1.29	-7.64	-4.07
DOQchk06	0.08	-0.57	-0.67	-0.93	-0.83	-1.95
DOQchk07	-0.61	0.07	-0.05	0.08	-1.37	-1.15
DOQchk08	-1.61	-0.24	-0.82	0.34	-1.47	0.40
DOQchk09	-4.28	-1.90	-6.07	-3.32	-6.62	-6.16
DOQchk10	-2.26	1.13	-0.87	1.24	-3.15	-0.49
DOQchk11	3.09	2.03	4.00	2.09	2.28	0.65
DOQchk12	-2.31	2.74	-2.45	2.73	-2.84	2.08
DOQchk13	0.47	-4.03	0.30	-4.48	-0.81	-6.27
DOQchk14	-3.49	-0.17	-5.05	-1.14	-5.12	-2.97
DOQchk15	1.65	-4.43	2.07	-4.32	1.12	-5.15
DOQchk16	-5.08	-3.05	-6.13	-3.20	-5.51	-3.37
DOQchk17	-1.27	-1.05	-3.40	-2.00	-2.66	-3.27
DOQchk18	6.18	-0.15	6.22	0.18	6.18	0.28
DOQchk19	0.30	-1.18	-2.13	-2.60	-1.80	-4.85
DOQchk20	-3.79	-2.97	-5.11	-3.43	-4.64	-4.15
DOQchk21	-0.95	-3.09	-3.96	-4.49	-2.78	-6.12
DOQchk22	-5.29	-5.77	-8.04	-7.17	-7.21	-9.04
DOQchk23	1.31	0.53	0.82	0.75	1.32	1.06
RMSE	3.77	2.51	4.20	3.00	4.31	4.22

Table B.6. Differences between known and computed coordinates of 30 checkpoints when using modified matching technique. Aerial Image coordinates are computed through navigation sensors data.

Check Point	Correlation > 0.60		Correlation > 0.65		Correlation > 0.70	
	dx	dy	dx	dy	dx	dy
GPSchk05	-3.20	4.52	-5.81	-0.27	-8.57	0.39
GPSchk06	3.78	-0.75	1.64	-4.32	-0.33	-3.74
GPSchk07	4.89	8.12	3.01	4.89	1.29	4.88
GPSchk08	-15.13	-12.77	-16.45	-13.44	-16.60	-11.35
GPSchk11	3.28	-10.44	1.47	-12.60	0.37	-10.92
GPSchk18	-0.06	-5.00	-0.81	-4.68	-0.27	-3.61
GPSchk21	-8.23	-7.84	-9.82	-9.54	-10.62	-8.09
DOQchk01	-1.70	3.71	-4.03	-0.44	-6.37	0.01
DOQchk02	-2.05	9.21	-4.85	3.96	-7.91	4.69
DOQchk03	-6.39	-10.86	-7.52	-11.10	-7.38	-9.15
DOQchk04	-10.82	-13.74	-11.81	-13.66	-11.47	-11.81
DOQchk05	-14.19	-10.70	-15.12	-10.48	-14.68	-8.68
DOQchk06	-0.25	-2.68	-0.78	-2.00	0.00	-1.36
DOQchk07	-1.24	-0.30	-3.04	-2.84	-4.35	-1.95
DOQchk08	3.89	5.73	2.30	3.29	1.09	3.34
DOQchk09	-13.67	-14.46	-13.83	-12.23	-12.09	-10.46
DOQchk10	-4.72	0.24	-7.45	-4.51	-10.21	-3.23
DOQchk11	1.73	1.48	-0.46	-1.99	-2.38	-0.94
DOQchk12	-0.60	3.57	-1.57	2.98	-1.60	3.43
DOQchk13	-3.14	-8.44	-4.46	-9.53	-4.84	-8.33
DOQchk14	-8.09	-7.70	-8.14	-5.52	-6.41	-4.40
DOQchk15	2.81	-3.28	1.27	-5.28	0.32	-4.67
DOQchk16	-1.68	-1.96	-1.66	-0.16	-0.13	-0.11
DOQchk17	-3.55	-7.21	-2.89	-3.37	-0.06	-2.67
DOQchk18	11.47	4.43	10.59	3.82	10.56	3.78
DOQchk19	-6.68	-12.37	-6.06	-8.33	-3.12	-7.00
DOQchk20	-3.07	-4.93	-2.97	-2.75	-1.21	-2.36
DOQchk21	-5.40	-12.70	-4.05	-6.99	-0.02	-6.12
DOQchk22	-10.72	-15.94	-9.68	-10.95	-6.11	-9.90
DOQchk23	7.23	4.94	6.90	5.66	7.75	5.44
RMSE	6.91	8.29	7.21	7.28	7.15	6.29

APPENDIX C
SOFTWARE LISTINGS

In this Appendix, the 'TEMPLATE MATCHING' software, written using Borland C++ Builder version 5.0, is listed. This application was written by the author and considered the main application utilized in this research. A Graphical User Interface (GUI) was designed to accept user input and processing parameters such as corelation search dimension, correlation threshold, used focal length. The application GUI was previously shown in Figure 4.8. Once the user enters the parameters and executes the application, the TEMPLATE MATCHING' program starts to read the input data and prepares the data for matching. The matched images are read and defined as objects of an 'IMAGE' class. In this class, many member functions were provided to perform correlation and least squares matching, image display, output matrices in bitmap format, etc. In the following a list of both the 'TEMPLATE MATCHING' application and the 'IMAGE' class is provided.

TEMPLATE MATCHING Application

```
/*
Author:     Amr Abd-Elrahman
Date:       04/05/2001
Objective:  Template Matching
GUI:        TempMatch Form
Input:      1. Reference Image (ERDAS gis format)
            2. ASCII file with Template images data(names, approx positions
               altitude horiz and vertical strip widths).
            3. ASCII file name for matching output results.
            4. Select Correlation and/or Least squares Matching.
            5. Select Pre-resampled template images or resample within least
               squares matching.
Output:     ASCII file containing matching results:
            image name, correlation matching results, correlation coefficient
            least squares matching results
*/
//TemplateMatch.h
//---------------------------------------------------------------------------
```

```
#ifndef TempMatchH
#define TempMatchH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Buttons.hpp>
#include <ExtCtrls.hpp>
#include <stdio.h>
#include <dos.h>
#include "ImgManip.h"
#include "ErdasFile.h"
#include <ComCtrls.hpp>
#include <jpeg.hpp>

//---------------------------------------------------------------------------

class TTempMatchingForm : public TForm
{

__published:          // IDE-managed Components
   TEdit *InputRefFileName;
   TBitBtn *InputRefFileButton;
   TEdit *InputTempFileName;
   TBitBtn *InputTempFileButton;
   TLabel *Label3;
   TLabel *Label4;
   TBitBtn *Start;
   TBitBtn *Cancel;
   TBitBtn *InputAerialImagesInfoButton;
   TEdit *InputImagesInformationFileName;
   TEdit *OutputMatchingResults;
   TBitBtn *OutputMatchingResultsBuuton;
   TLabel *Label1;
   TLabel *Label2;
   TCheckBox *CorrelationMatchingFlag;
   TLabel *Label5;
   TCheckBox *LSMatchingFlag;
   TCheckBox *ResampledChkBox;
   TCheckBox *ElimStripFlag;
     TBevel *Bevel2;
     TBevel *Bevel3;
     TEdit *CorrThresholdEdit;
     TEdit *SearchWidthEdit;
     TEdit *SearchDepthEdit;
     TLabel *Label6;
     TLabel *Label7;
     TLabel *Label8;
     TLabel *Label9;
     TEdit *FocalLengthEdit;
     TBevel *Bevel1;
     TLabel *Label11;
     TLabel *Label10;
     TProgressBar *ProgressBar1;
     TEdit *PixSizeEdit;
     TLabel *Label12;
   void __fastcall InputRefFileButtonClick(TObject *Sender);
   void __fastcall InputTempFileButtonClick(TObject *Sender);
   void __fastcall StartClick(TObject *Sender);
   void __fastcall OutputMatchingResultsBuutonClick(TObject *Sender);
   void __fastcall InputAerialImagesInfoButtonClick(TObject *Sender);


private:    // User declarations

void ReadTempImage();
void ReadRefImage();
void CreateApproxUnknowns();
void WriteMatchResults();
void GetImgParameters();

unsigned char* buffer;
double PixSize,RScale,f,Scale,X0,Y0,Xi,Yi,r0,c0,rMatch,cMatch, SatRes
      , xcomp,ycomp, CorrThresh;
int win, ResampHeight, ResampWidth, rtol, ctol, TempCounter, HLstrip, VLstrip;
Image *TempImg;
Image *RefImg;

FILE* ErdasFileHandler;
struct ErdasHeader{
 char hdword[6];
 short ipack, nbands;
 char unused1[6];
 long icols, irows, xstart, ystart;
 char unused2[56];
 short maptyp, nclass;
 char unused3[14];
 short iautyp;
 float acre, xmap, ymap, xcell, ycell;
} ErdasImgHeader;

struct ImagesData{
            int id;
            char name[100];
   float x;
   float y;
   float H;
   float A;
   int OF;
   int Hstrip;
   int Vstrip;
  };

struct OutputData{
            int id;
   double a0,a1,a2,b0,b1,b2,c1,c2;
   double rMatch,cMatch;
}ImgMatchData;
FILE *ImgInfoFile,*ImgOutputFile;
ImagesData *ImgsData;
int TempImagesNo;
int LSFlag;
long rul,cul,rlr,clr, a0Corr, b0Corr;
double aInit,bInit;
struct time t;
float CorrStart,CorrEnd;
char ImgName[100];
public:                  // User declarations
   __fastcall TTempMatchingForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TTempMatchingForm *TempMatchingForm;
//---------------------------------------------------------------------------
#endif
```

//TemplateMatch.cpp

```
//---------------------------------------------------------------------------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <math.h>
#include "TempMatch.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#define PI 3.14159265358979
TTempMatchingForm *TempMatchingForm;
//---------------------------------------------------------------------------
```

```
__fastcall TTempMatchingForm::TTempMatchingForm(TComponent* Owner)
   : TForm(Owner)
{


}
//---------------------------------------------------------------------------
void __fastcall TTempMatchingForm::InputRefFileButtonClick(TObject *Sender)
{

  //Select and read header for reference image
  TOpenDialog *OpenDlg1 = new TOpenDialog(this);
  OpenDlg1->DefaultExt = String("gis");
  OpenDlg1->FileName = String("*.gis");
  OpenDlg1->InitialDir = String("c:\\papers\\chk-pos-acc\\data");
  if (OpenDlg1->Execute())
  {
    InputRefFileName->SelText = ExtractFileName(OpenDlg1->FileName);
    ErdasFileHandler = fopen((OpenDlg1->FileName).c_str(), "rb");
    fread(&ErdasImgHeader, sizeof(ErdasHeader), 1, ErdasFileHandler );
  }

}
//---------------------------------------------------------------------------

// Read Individual template image file
void __fastcall TTempMatchingForm::InputTempFileButtonClick(TObject *Sender)
{
 long Width, Height;
  TOpenDialog *OpenDlg1 = new TOpenDialog(this);
  OpenDlg1->DefaultExt = String("bmp");
  OpenDlg1->FileName = String("*.bmp");
  OpenDlg1->InitialDir = String("c:\\PhD\\phd-analysis\\data");
  if (OpenDlg1->Execute())
  {
    InputTempFileName->SelText = ExtractFileName(OpenDlg1->FileName);
    try
    {
      Graphics::TBitmap *TempBitmap = new Graphics::TBitmap();
      TJPEGImage *Tempjpg = new TJPEGImage();
      Tempjpg->LoadFromFile(OpenDlg1->FileName);
      Width = Tempjpg->Width;
      Height = Tempjpg->Height;
      TempBitmap->Assign(Tempjpg);
      TempImg = new Image(Width, Height);
      for (int i = 0 ; i <Height; i++)
       for (int j = 0; j < Width; j++)
           TempImg->ImageMatrix[i][j] = TempBitmap->Canvas->Pixels[j][i];
      delete TempBitmap;
      TempImg->CreateBitMap();
      TempImg->DisplayBitMap();
    }
    catch (...)
    {
      Application->MessageBox("Error in Opening Template File", "Message Box", MB_OKCANCEL);
    }

  }
}

//---------------------------------------------------------------------------

// Read Individual template Images.
void TTempMatchingForm::ReadTempImage()
{
 long Width, Height;
 HLstrip =  ImgsData[TempCounter].Hstrip;
 VLstrip =  ImgsData[TempCounter].Vstrip;
    try
    {
```

```
    Graphics::TBitmap *TempBitmap = new Graphics::TBitmap();
    String ImgNme = String(ImgsData[TempCounter].name);

    if ((ImgNme.SubString(ImgNme.Length()-2,3)).LowerCase() == "jpg")
     {
      TJPEGImage *Tempjpg = new TJPEGImage();
      Tempjpg->LoadFromFile(ImgsData[TempCounter].name);
      Width = Tempjpg->Width - VLstrip;
      Height = Tempjpg->Height - HLstrip;
      TempBitmap->Assign(Tempjpg);
      delete Tempjpg;
     }
    else
     {
      TempBitmap->LoadFromFile(ImgsData[TempCounter].name);
      Width = TempBitmap->Width - VLstrip;
      Height = TempBitmap->Height - HLstrip;
     }
    if (ElimStripFlag->State ==  cbUnchecked)
    {
       TempImg = new Image(Width, Height);
       for (int i = 0 ; i <Height; i++)
        for (int j = 0; j < Width; j++)
           TempImg->ImageMatrix[i][j] = TempBitmap->Canvas->Pixels[j][i];
        delete TempBitmap;
    }
     else if (ImgsData[TempCounter].OF == 1)
     {
       TempImg = new Image(Width, Height);
       for (int i = 0 ; i <Height; i++)
        for (int j = 0; j < Width; j++)
           TempImg->ImageMatrix[i][j] = TempBitmap->Canvas->Pixels[j][i+HLstrip];
        delete TempBitmap;
     }
     else
     {
       TempImg = new Image(Width, Height);
       for (int i = 0 ; i <Height; i++)
        for (int j = 0; j < Width; j++)
           TempImg->ImageMatrix[Height - i - 1][Width - j - 1] = TempBitmap->Canvas->Pixels[j][i +HLstrip];
        delete TempBitmap;
     }
    }
   catch (...)
    {
    Application->MessageBox("Error in Opening Template File", "Message Box", MB_OKCANCEL);
    }

}

//---------------------------------------------------------------------------------------------------------------------
// Process template image data to identify search area for correlation matching
void TTempMatchingForm::GetImgParameters()
{

   double xmap,ymap,dyMax,dxMax;
   double rot,rotabs;
   xmap = ErdasImgHeader.xmap;
   ymap = ErdasImgHeader.ymap;
   rotabs = abs(ImgsData[TempCounter].A)*PI/180;
   rot = ImgsData[TempCounter].A *PI/180;
   SatRes = ErdasImgHeader.xcell;

    if (ResampledChkBox->State ==  cbUnchecked)
     {
     //case of using resampled aerial images
       if (ImgsData[TempCounter].OF > 0)
          {

             // case of existing blank strips in the image
        xcomp =  ((TempImg->ImgWidth + VLstrip)* cos(rot) -(TempImg->ImgHeight - HLstrip)* sin(rot))*SatRes/2;
        ycomp =  ((TempImg->ImgHeight - HLstrip) * cos(rot) +(TempImg->ImgWidth + VLstrip) * sin(rot))*SatRes/2;
        dxMax =  ((TempImg->ImgWidth + VLstrip) * cos(rotabs) - (TempImg->ImgHeight - HLstrip) * sin(rotabs))*SatRes/2;
        dyMax =  ((TempImg->ImgHeight - HLstrip) * cos(rotabs) + (TempImg->ImgWidth + VLstrip)* sin(rotabs))*SatRes/2;
        X0 = ImgsData[TempCounter].x - xcomp;
        Y0 = ImgsData[TempCounter].y + ycomp;

        // correcting satellite coordinates due to shift
        r0 = ((ErdasImgHeader.ymap - 765)-Y0)/SatRes;
        c0 = (X0 - (ErdasImgHeader.xmap + 150))/SatRes;
        rul = r0 - rtol;
        cul = c0 - ctol;
        rlr = r0 + dyMax/SatRes*2 + rtol;
        clr = c0 + dxMax/SatRes*2 + ctol;
     }
   else
     {

       // case of existing blank strips in the image
       xcomp =  ((TempImg->ImgWidth - VLstrip)* cos(rot) -(TempImg->ImgHeight + HLstrip)* sin(rot))*SatRes/2;
       ycomp =  ((TempImg->ImgHeight + HLstrip) * cos(rot) +(TempImg->ImgWidth - VLstrip) * sin(rot))*SatRes/2;
       dxMax =  ((TempImg->ImgWidth - VLstrip) * cos(rotabs) - (TempImg->ImgHeight + HLstrip) * sin(rotabs))*SatRes/2;
       dyMax =  ((TempImg->ImgHeight + HLstrip) * cos(rotabs) + (TempImg->ImgWidth - VLstrip)* sin(rotabs))*SatRes/2;
       X0 = ImgsData[TempCounter].x - xcomp;
       Y0 = ImgsData[TempCounter].y + ycomp;

       // correcting satellite coordinates due to shift
       r0 = ((ErdasImgHeader.ymap - 765)-Y0)/SatRes;
       c0 = (X0 - (ErdasImgHeader.xmap + 150))/SatRes;
       rul = r0 + dyMax/SatRes*2 - rtol;
       cul = c0 + dxMax/SatRes*2 - ctol;
       rlr = r0 + rtol;
       clr = c0 + ctol;
      }

 }
else
 {
 //case of no pre-resampling aerial images
 Scale = f / ImgsData[TempCounter].H;
 RScale = PixSize/SatRes;
 win = SatRes*Scale/PixSize;
 ResampHeight = TempImg->ImgHeight/win;
 ResampWidth = TempImg->ImgWidth/win;
 if (fmod(win,2) == 0)
 win = win -1;
  if (ImgsData[TempCounter].OF > 0)
 {

  xcomp =  ((TempImg->ImgWidth + VLstrip)* cos(rot) -(TempImg->ImgHeight - HLstrip)* sin(rot))*PixSize/Scale/2;
  ycomp =  ((TempImg->ImgHeight - HLstrip) * cos(rot) +(TempImg->ImgWidth + VLstrip) * sin(rot))*PixSize/Scale/2;
  dxMax =  ((TempImg->ImgWidth + VLstrip) * cos(rotabs) - (TempImg->ImgHeight - HLstrip) * sin(rotabs))*PixSize/Scale/2;
  dyMax =  ((TempImg->ImgHeight - HLstrip) * cos(rotabs) + (TempImg->ImgWidth + VLstrip)* sin(rotabs))*PixSize/Scale/2;

  X0 = ImgsData[TempCounter].x - xcomp;
  Y0 = ImgsData[TempCounter].y + ycomp;

  // correcting satellite coordinates due to shift
  r0 = ((ErdasImgHeader.ymap - 765)-Y0)/SatRes;
  c0 = (X0 - (ErdasImgHeader.xmap + 150))/SatRes;

  rul = r0 - rtol;
  cul = c0 - ctol;
  rlr = r0 + dyMax/SatRes*2 + rtol;
  clr = c0 + dxMax/SatRes*2 + ctol;
 }
 else
 {
  xcomp =  ((TempImg->ImgWidth - VLstrip)* cos(rot) -(TempImg->ImgHeight + HLstrip)* sin(rot))*PixSize/Scale/2;
  ycomp =  ((TempImg->ImgHeight + HLstrip) * cos(rot) +(TempImg->ImgWidth - VLstrip) * sin(rot))*PixSize/Scale/2;
```

```
      dxMax =  ((TempImg->ImgWidth - VLstrip) * cos(rotabs) - (TempImg->ImgHeight + HLstrip) * sin(rotabs))*PixSize/Scale/2;
      dyMax =  ((TempImg->ImgHeight + HLstrip) * cos(rotabs) + (TempImg->ImgWidth - VLstrip)* sin(rotabs))*PixSize/Scale/2;
      X0 = ImgsData[TempCounter].x - xcomp;
      Y0 = ImgsData[TempCounter].y + ycomp;

      // correcting satellite coordinates due to shift
      r0 = ((ErdasImgHeader.ymap - 765)-Y0)/SatRes;
      c0 = (X0 - (ErdasImgHeader.xmap + 150))/SatRes;

      rul = r0 + dyMax/SatRes*2 - rtol;
      cul = c0 + dxMax/SatRes*2 - ctol;
      rlr = r0 + rtol;
      clr = c0 + ctol;
     }
   }

}
//---------------------------------------------------------------------------
// Read Reference image. Different area is loaded for each template image. This
// avoids reading and processing the whole reference image.
void TTempMatchingForm::ReadRefImage()
{

   long Height, Width;
   Height = ErdasImgHeader.irows;
   Width = ErdasImgHeader.icols;

 // Allocate buffer //
 //buffer = malloc( Width * sizeof(unsigned char) );
  buffer = new unsigned char[Width];
  RefImg = new Image(Width, rlr-rul);
              /* Loop through file to read and display rows */
   fseek(ErdasFileHandler,(rul-1)*Width + sizeof(ErdasHeader),SEEK_SET);
              for (int i=0; i<(rlr-rul); i++) {
        fread( buffer, Width, 1, ErdasFileHandler );
        for(int j = 0; j < Width; j++) {
           RefImg->ImageMatrix[i][j] = *(buffer + j);
         }
       }

}
//---------------------------------------------------------------------------
// Create approximate values for affine transformation parameters between
// matched images.
void TTempMatchingForm::CreateApproxUnknowns()
{
 double Azimuth,scale;
 Azimuth = ImgsData[TempCounter].A * PI/180;
 scale = 1.0;
 aInit = r0-rul;
 bInit = c0;
 RefImg->a0 = r0-rul;
 RefImg->a1 = cos(Azimuth)*scale;
 RefImg->a2 = sin(Azimuth)*scale;
 RefImg->b0 = c0;
 RefImg->b1 = -sin(Azimuth)*scale;
 RefImg->b2 = cos(Azimuth)*scale;
 RefImg->c1 = 0;
 RefImg->c2 = 0;
 RefImg->rul = rul;

}
//---------------------------------------------------------------------------
// start execution
void __fastcall TTempMatchingForm::StartClick(TObject *Sender)
{

 f = ((FocalLengthEdit->Text).ToDouble())/1000;
 PixSize = ((PixSizeEdit->Text).ToDouble())/1000000;
 CorrThresh = (CorrThresholdEdit->Text).ToDouble();
```

```
  ctol = (SearchWidthEdit->Text).ToInt();
  rtol = (SearchDepthEdit->Text).ToInt();
  ProgressBar1->Min = 0;
  ProgressBar1->Max = 100;

/*  f = 0.020;
  PixSize = .000009;
  rtol = 150;
  ctol = 150;
*/
  LSFlag = 0;
  TempMatchingForm->Update();
// String nme;
 for (TempCounter = 0 ; TempCounter <TempImagesNo; TempCounter++)
   {
    String ImgNme = String(ImgsData[TempCounter].name);
    strcpy(ImgName, (ImgNme.SubString(ImgNme.LastDelimiter("//") + 1, (ImgNme.Length()- ImgNme.LastDelimiter("//") -
4))).c_str());
    // call function to read template image
    ReadTempImage();
    // call function that process template image data and define search area
    GetImgParameters();
    // read corresponding part of reference image
    ReadRefImage();
    // Compute approximate values for the unknowns (neded for least squares matching)
    CreateApproxUnknowns();
    //case of using Resampled Images
    if (ResampledChkBox->State ==  cbUnchecked)
    {
     // Case Least Squares matching Only
     if (CorrelationMatchingFlag == cbUnchecked)
      {
      RefImg->CorrMax = 0;
      RefImg->PerformLSMatching(TempImg, LSFlag, ImgName);
      WriteMatchResults();
      }
     //case correlation matching followd by least squares matching
     else
      {

      //gettime(&t);
      //CorrStart = t.ti_hour*3600 +  t.ti_min * 60 + t.ti_sec + (float)t.ti_hund/100;
      RefImg->PerformCorrMatching(TempImg,rtol,ctol);
      //gettime(&t);
      //CorrEnd = t.ti_hour*3600 +  t.ti_min * 60 + t.ti_sec + (float)t.ti_hund/100;
      if (RefImg->CorrMax > CorrThresh)
      {
       a0Corr = RefImg->a0;
       b0Corr = RefImg->b0;
       RefImg->PerformLSMatching(TempImg, LSFlag, ImgName);
      }
      WriteMatchResults();

      }

    }
    //case of Resampling with matching
    else
    {
     // Case Least Squares matching Only
     if (CorrelationMatchingFlag == cbUnchecked)
      {
      RefImg->CorrMax = 0;
      RefImg->PerformLSResampDiagMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth, ImgName);
      //RefImg->PerformLSResampOneMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth, ImgName);
      //RefImg->PerformLSResampSymMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth,ImgName );
      //RefImg->PerformLSResampMatch(TempImg, LSFlag, 13, 31, 47);
       WriteMatchResults();
      }
     //case correlation matching followd by least squares matching
```

```
      else
      {
       RefImg->PerformResampCorrMatching(TempImg,win,ResampHeight,ResampWidth,rtol,ctol);
       a0Corr = RefImg->a0;
       b0Corr = RefImg->b0;

      // RefImg->PerformLSResampCircularMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth, ImgName);
       //RefImg->PerformLSResampDiagOverlapMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth, ImgName);
//RefImg->PerformLSResampOneMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth, ImgName);
      if (RefImg->CorrMax > CorrThresh)
       {
         RefImg->PerformLSResampDiagMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth, ImgName);
      //RefImg->PerformLSResampSymMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth,ImgName );
       //RefImg->PerformLSResampOctMatch(TempImg, LSFlag, win, ResampHeight, ResampWidth,ImgName );
      //RefImg->PerformLSResampMatch(TempImg, LSFlag, 13, 31, 47);
      // WriteMatchResults();
       }
       WriteMatchResults();

      }
    }
    ProgressBar1->Position = (TempCounter*100)/TempImagesNo;
    TempMatchingForm->Update();
    ProgressBar1->Update();
    delete TempImg;
    delete RefImg;
  }
fclose(ErdasFileHandler);
fclose(ImgOutputFile);

}
//-----------------------------------------------------------------------------------------------------------------------------
// Read Matching Results output File Name
void __fastcall TTempMatchingForm::OutputMatchingResultsBuutonClick(
     TObject *Sender)
{

 TOpenDialog *OpenDlg1 = new TOpenDialog(this);
 OpenDlg1->DefaultExt = String("txt");
 OpenDlg1->FileName = String("*.txt");
 OpenDlg1->InitialDir = String("c:\\phd\\phd-analysis\\output");
 if (OpenDlg1->Execute())
 {
  OutputMatchingResults->SelText = ExtractFileName(OpenDlg1->FileName);
  ImgOutputFile = fopen((OpenDlg1->FileName).c_str(), "wt"); /* open file TEST.$$$ */

 }

}
//-----------------------------------------------------------------------------------------------------------------------------
// Write Matching Results to Output File

void TTempMatchingForm::WriteMatchResults()
{
 double xc,yc, a0, a1, a2, b0, b1, b2, Xc, Yc, WResampFactor, HResampFactor;
 int Height, Width;
 Height = TempImg->ImgHeight;
 Width = TempImg->ImgWidth;
 a0 = RefImg->a0;
 a1 = RefImg->a1;
 a2 = RefImg->a2;
 b0 = RefImg->b0;
 b1 = RefImg->b1;
 b2 = RefImg->b2;

 if (ResampledChkBox->State == cbUnchecked)

   WResampFactor = HResampFactor = 1.0;
 else
  {
   WResampFactor = RefImg->JumpW;
   HResampFactor = RefImg->JumpH;
  }

 if (ImgsData[TempCounter].OF > 0)
 {
   xc = (rul - 1) + a0 + a1 * (Height - HLstrip-HResampFactor)/2/HResampFactor + a2 * (Width + VLstrip-
WResampFactor)/2/WResampFactor;
   yc =  b0 + b1 * (Height - HLstrip-HResampFactor) /2/HResampFactor + b2 * (Width + VLstrip-
WResampFactor)/2/WResampFactor;
 }
 else
 {
   xc = rul + a0 + a1 * (Height + HLstrip-HResampFactor) /2/HResampFactor + a2 * (Width - VLstrip-
WResampFactor)/2/WResampFactor;
   yc =  b0 + b1 * (Height + HLstrip-HResampFactor) /2/HResampFactor + b2 * (Width - VLstrip-
WResampFactor)/2/WResampFactor;
 }
 Xc = ErdasImgHeader.xmap + yc * SatRes;
 Yc = ErdasImgHeader.ymap - xc * SatRes;

 rMatch = r0 + RefImg->rMatch;
 cMatch = c0 + RefImg->cMatch;

 fprintf(ImgOutputFile, "%d %s %f %f %f %f %f %f %f %f %d %d %f %f %f \n", ImgsData[TempCounter].id,
       ImgsData[TempCounter].name, (RefImg->a0+rul), RefImg->a1, RefImg->a2, RefImg->b0,
       RefImg->b1, RefImg->b2, aInit+rul, bInit,a0Corr + rul, b0Corr, RefImg->CorrMax, Xc, Yc);

}
//---------------------------------------------------------------------------------------------------------------------
// Read Template images Data File
void __fastcall TTempMatchingForm::InputAerialImagesInfoButtonClick(
     TObject *Sender)
{
  int ImagesNo = 300;
  try
   {
    ImgsData = new ImagesData[ImagesNo];
   }
  catch (...)
   {
    Application->MessageBox("Error allocating memory", "Message Box", MB_OKCANCEL);
   }
  TOpenDialog *OpenDlg1 = new TOpenDialog(this);
  OpenDlg1->DefaultExt = String("txt");
  OpenDlg1->FileName = String("*.txt");
  OpenDlg1->InitialDir = String("c:\\phd\\phd-analysis\\data");
  if (OpenDlg1->Execute())
  {
   InputImagesInformationFileName->SelText = ExtractFileName(OpenDlg1->FileName);
   ImgInfoFile = fopen((OpenDlg1->FileName).c_str(), "rt"); /* open file TEST.$$$ */
   int size = sizeof(ImagesData);
   int i = 0;
   char TempImgName[100];
   while(!feof(ImgInfoFile))
    {
     int numread = fscanf(ImgInfoFile, "%d %s %f %f %f %f %d %d %d", &ImgsData[i].id, ImgsData[i].name,
                    &ImgsData[i].x, &ImgsData[i].y,&ImgsData[i].H, &ImgsData[i].A, &ImgsData[i].OF,
                    &ImgsData[i].Hstrip,&ImgsData[i].Vstrip  );
     i++;
    }
   fclose(ImgInfoFile); /* close file */
   TempImagesNo = i-1 ;
  }

}
//---------------------------------------------------------------------------------------------------------------------
```

IMAGE Class

```
/* Author:     Amr Abd-Elrahman
Date:     01/05/2001
Objective:  Image class with member functions for displaying, saving,
         matching, and filtering images
GUI:        None
*/
//ImageManip.h
//---------------------------------------------------------------------------
#ifndef imgManipForList_hH
#define imgManipForList_hH
//---------------------------------------------------------------------------
#include   <stdio.h>
#include   <stdlib.h>
#include   <dos.h>
class Image{

//---------------------------------------------------------------------------
public:
 Image(long, long);
 ~Image();
 void CreateBitMap();
 void DisplayBitMap();
 void SaveBitMap();
 //LeastSquares

 void PerformLSResampCircularMatch(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW,
                                         char *TempNme);
 void PerformLSResampDiagMatch(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char *TempNme);
 void PerformLSResampSymMatch(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char *TempNme);
 void PerformLSMatching(Image *TImg,int LSFlag, char *ImgNme);
 void PerformResampCorrMatching(Image *TImg, int ResampWinSize, int ResampH, int ResampW);
 void PerformCorrMatching(Image *TImg);
 void PerformGaussianFilter(int Gwin, float sigma);
long ImgWidth, ImgHeight;
 float JumpH, JumpW;
 unsigned char **ImageMatrix;
 Graphics::TBitmap* ImageBitmap;
 double h0, h1, a0, a1, a2, b0, b1, b2,c1,c2,rul, xr, yr,rMatch,cMatch,a3,a4,a5,a6, CorrMax;
 double *Unknowns;

private:
void SetAffineLSResampCircularMatchInitialValues();
void SetAffineLSResampDiagMatchInitialValues();
void SetAffineLSResampSymMatchInitialValues();
void SetCommonParameters(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char *TempNme);
void SetAffineLSInitialValues();
void LSAffineResampCircularMatch();
void LSAffineResampDiagMatch();
void LSAffineResampOneMatchNew();
void LSAffineResampSymMatch();
void LSAffineMatch();
int  GetMaxCorr(int winParNo);
void GetUnknownsVCMtrix();
void MatrixInverse(double **, int);
void choldc();
void cholsl();
unsigned char BilinearResamp(float x, float y);
void AdjustSymObs();
void AdjustObs();
void GetVariance();
void AdjustUnknowns();
void WriteLSResults();
void CloseOutputFiles();
void OpenOutputFiles(char * TempNme);
void FreeMemory();
void AdjustWeights();
void GetQwinUnknownsArray();
//Data Members For Least Squares Matching
Image *TempImg,*PyramidImage, *OriginalImage;
char * TempName;
int iterations, LSTypeFlag, win, Qwin,cQwin,Owin,ResampHeight, ResampWidth, vmax, vmin,vthresh, Gwin, UnknownsNo ;
long N, CondNo; //no of equations
double *l,**W, WMatrix;
double **M, **MM, **lM;//BT * B
double *U ;/
double *corr;// solution vector (corr to parameters)
double *p;//needed to store diagonal wehen decomposing using Cholesky method
double **winUnknowns, *QwinUnknownsArray;
double **B, **AA,**AAinv,**Temp, *f, *v,item, *Q;
int *PixMap;
long double RefVar, RefVarI ,OldVar;
long r,c, rl, cl, xl, yl, ind,i,j,k;
FILE *stream, *streamI;
long TempHeight, TempWidth;
//Data Members For Correlation Matching
Image* CorrImg;
long rMax, cMax, rr, cc, r0, c0;
int rowsNo, colsNo;
double ** CorrMatrix, SigmaX, SigmaY, SigmaXY, SigmaX2, SigmaY2, tilt, swing, azimuth;
double SXY, SXX, SYY, Omega, Phi, Kappa;
struct time t;
float Start,End;
};
#endif
//---------------------------------------------------------------------------------------------------------------------------------------------------
```

```
//ImageManip.cpp
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include  "ImgManip.h"
#include  <math.h>
#include  <stdio.h>
#include  <stdlib.h>
#include  <alloc.h>
#include  <conio.h>
#include  <vcl.h>
#include  <values.h>
#include  <dos.h>
#include  <string.h>

#define iterationsThreshold 0.0009999
#define Maxiterations 40

//---------------------------------------------------------------------------
#pragma package(smart_init)
// Image Class Instructor
//Creates a 2-d matrix with image grey level values
Image::Image(long w, long h)
{
 ImgWidth = w;
 ImgHeight = h;
 ImageMatrix = new unsigned char*[ImgHeight];
  for (int i=0; i < ImgHeight; i++)
    ImageMatrix[i] = new unsigned char[ImgWidth];
 }

// Image Class destructor
Image::~Image()
{
//delete Imagematrix;
 for (int i = 0; i < ImgHeight ; i++)
    delete[] ImageMatrix[i];
delete[] ImageMatrix;
}
```

```
//--------------------------------------------------------------------------
// Create bitmap from a 2-dimensional matrix
void Image::CreateBitMap()
{

ImageBitmap = new Graphics::TBitmap;
ImageBitmap->Width = ImgWidth;
ImageBitmap->Height = ImgHeight;


for (int i =0; i < ImgHeight; i++)
 for (int j = 0; j < ImgWidth; j++)
  {

   int pixel =   ImageMatrix[i][j];
   ImageBitmap->Canvas->Pixels[j][i] = RGB(pixel, pixel, pixel);
  }
}

//--------------------------------------------------------------------------
// save images created as bitmaps
void Image::SaveBitMap()
{
 TSaveDialog *SaveDialog = new TSaveDialog(NULL);
 if (ImageBitmap)
  if (SaveDialog->Execute())
   {
    ImageBitmap->SaveToFile(SaveDialog->FileName);
    delete SaveDialog;
   }
 else
   Application->MessageBox("No BitMap Created","BitMap Save Error",MB_OKCANCEL);
}

//Least Squares Matching
//--------------------------------------------------------------------------
//Start LS matching technique with Circular symmetry assumption
void Image::PerformLSResampCircularMatch(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char
*TempNme)
{

 // call function to Set comon Parameters
 SetCommonParameters(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char *TempNme)
  // call function to initialize matrices
 SetAffineLSResampCircularMatchInitialValues();
 // Apply Least Squares Matching
 LSAffineResampCircularMatch();
 //close open output ASCII files
 CloseOutputFiles();
}


//--------------------------------------------------------------------------
//Start LS matching technique with Diagonal symmetry assumption
void Image::PerformLSResampDiagMatch(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char
*TempNme)
{

 // call function to Set comon Parameters
  SetCommonParameters(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char *TempNme)
  // call function to initialize matrices
 SetAffineLSResampDiagMatchInitialValues();
 // Apply Least Squares Matching
 LSAffineResampDiagMatch();
 //close open output ASCII files
 CloseOutputFiles();
}

//--------------------------------------------------------------------------
//Start LS matching technique with four-quarter symmetry assumption
```

```
void Image::PerformLSResampSymMatch(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char
*TempNme)
{

 // call function to Set comon Parameters
  SetCommonParameters(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char *TempNme)
  // call function to initialize matrices
 SetAffineLSResampSymMatchInitialValues();
 // Apply Least Squares Matching
 LSAffineResampSymMatch();
 //close open output ASCII files
 CloseOutputFiles();

}
//------------------------------------------------------------
//initialize output Files and resampling spacings
void Image::SetCommonParameters(Image *TImg, int LSFlag, int ResampWinSize, int ResampH, int ResampW, char *TempNme)
{
 //Initialize  2 Output ASCII files for individual images listing:
 //1. matching results
 //2. Output filter parameters

 OpenOutputFiles(TempNme);
 // Compute resampling spacing in both x and y directions
 float Jump;
 TempImg = TImg;
 win = ResampWinSize;
 ResampHeight = ResampH;
 ResampWidth = ResampW;
 JumpW =  (float)TempImg->ImgWidth / ResampWidth;
 JumpH = (float)TempImg->ImgHeight / ResampHeight;
 if (JumpH < JumpW)
   Jump = JumpH;
 else Jump = JumpW;
 if (win >= Jump)
    win = (int)(Jump-1);
 Qwin = win/2+1;

}
//------------------------------------------------------------
//Start Traditional (No resampling) LS matching
void Image::PerformLSMatching(Image *TImg, int LSFlag, char *TempNme)
{
TempName = strcat(TempNme,".txt");

if ((stream = fopen(TempName, "wt")) == NULL) /* open file TEST.$$$ */
   Application->MessageBox("File I/O error","Failure to Open Output File", MB_OKCANCEL);
fprintf(stream, " Least Squares Results \n");

LSTypeFlag = LSFlag;
TempImg = TImg;

  SetAffineLSInitialValues();
  LSAffineMatch();

fclose(stream); /* close file */
}

//----------------------------------------------------------------------------

//Declare and Initialize coefficient matrices for LS Matching with
//circular symmetry assumption
void Image::SetAffineLSResampCircularMatchInitialValues()
{

  UnknownsNo = Qwin+7-2;
  cQwin = Qwin -2;
  OldVar = RefVar = MAXDOUBLE;
  iterations = 0;
  CondNo = ResampHeight*ResampWidth;//no of equations
```

```
TempHeight = TempImg->ImgHeight;
TempWidth = TempImg->ImgWidth;

try
{
 // Initialize p temporarily vector for solving Normal equation system
 p = new double[CondNo];
 // Initialize vector for unknowns corrections
 corr = new double[UnknownsNo];
 // Initialize U, vector of normal equation system RHS
 U = new double[UnknownsNo];
 // Initialize vector of Unknowns
 Unknowns = new double[UnknownsNo];
 QwinUnknownsArray = new double[Qwin*Qwin];
 // Initialize normal equation System matrix
 M = new double*[UnknownsNo];
 for (int i = 0; i < UnknownsNo; i++)
   M[i] = new double[UnknownsNo];

 IM = new double*[TempHeight];
 for (int i = 0; i < TempHeight; i++)
   IM[i] = new double[TempWidth];

 // Unknowns parameter as window
 winUnknowns = new double*[win];
 for(int i = 0; i < win; i++)
   winUnknowns[i] = new double[win];
 // Temporarily Image Matrix
 MM = new double*[UnknownsNo];
 for (int i = 0; i < UnknownsNo; i++)
   MM[i] = new double[UnknownsNo];
 // Initialize Least Squares Matrices
 for(i = 0; i< UnknownsNo; i++)
   {
    U[i] = 0;
    corr[i]=0;
    for(j=0;j<UnknownsNo;j++)
      M[i][j]=0;
   }
 // Temporarily Image Matrix
 for (i=0; i < TempHeight; i++)
   for (j = 0; j<TempWidth; j++)
    IM[i][j] = (double)TempImg->ImageMatrix[i][j];

 N = ResampHeight*ResampWidth*win*win;
 // Initialize misclosure vector
 f = new double[CondNo];
 l = new double[CondNo];
 // Initialize residuals vector
 v = new double[N];
 // Initialize coefficient matrix
 B = new double*[CondNo];
 for (int i=0; i < CondNo; i++)
  B[i] = new double[UnknownsNo];

 }
 catch(...)
 {
  Application->MessageBox("Error Initializing Memory","Initialization failed", MB_OKCANCEL);
 }

for (i = 0; i <win;i++)
 for(j=0;j<win; j++)
  winUnknowns[i][j] = 0.0;
int winsq = win*win;
// Initial values for unknowns
for (i =0; i < cQwin; i++)
  Unknowns[i] = 1.0/winsq;
Unknowns[cQwin+0] = 0;
Unknowns[cQwin+1] = a0;
```

```
 Unknowns[cQwin+2] = a1;
 Unknowns[cQwin+3] = a2;
 Unknowns[cQwin+4] = b0;
 Unknowns[cQwin+5] = b1;
 Unknowns[cQwin+6] = b2;
 rl = 0;
 cl = 0;
 RefVar1 = 0.0;

}

//-----------------------------------------------------------------------------
////Declare and Initialize coefficient matrices for LS Matching with
//Diagonal symmetry assumption
void Image::SetAffineLSResampDiagMatchInitialValues()
{
 UnknownsNo = Qwin+7;
 OldVar = RefVar = MAXDOUBLE;
 iterations = 0;
 CondNo = ResampHeight*ResampWidth;//no of equations
 TempHeight = TempImg->ImgHeight;
 TempWidth = TempImg->ImgWidth;

 try
 {
  p = new double[CondNo];
  corr = new double[UnknownsNo];
  U = new double[UnknownsNo];
  Unknowns = new double[UnknownsNo];
  QwinUnknownsArray = new double[Qwin*Qwin];

  M = new double*[UnknownsNo];
  for (int i = 0; i < UnknownsNo; i++)
    M[i] = new double[UnknownsNo];

  IM = new double*[TempHeight];
  for (int i = 0; i < TempHeight; i++)
    IM[i] = new double[TempWidth];


  winUnknowns = new double*[win];
  for(int i = 0; i < win; i++)
    winUnknowns[i] = new double[win];

  MM = new double*[UnknownsNo];
  for (int i = 0; i < UnknownsNo; i++)
    MM[i] = new double[UnknownsNo];


  for(i = 0; i< UnknownsNo; i++)
    {
     U[i] = 0;
     corr[i]=0;
     for(j=0;j<UnknownsNo;j++)
       M[i][j]=0;
    }


  for (i=0; i < TempHeight; i++)
    for (j = 0; j<TempWidth; j++)
      IM[i][j] = (double)TempImg->ImageMatrix[i][j];


  N = ResampHeight*ResampWidth*win*win; //No of Obs

  f = new double[CondNo];
  I = new double[CondNo];
  v = new double[N];
  B = new double*[CondNo];
  for (int i=0; i < CondNo; i++)
   B[i] = new double[UnknownsNo];

 }
 catch(...)
 {
  Application->MessageBox("Error Initializing Memory","Initialization failed", MB_OKCANCEL);
 }

 for (i = 0; i <win;i++)
  for(j=0;j<win; j++)
   winUnknowns[i][j] = 0.0;
 int winsq = win*win;

 // initiate parameters for diagmatch
  for (i =0; i < Qwin; i++)
   Unknowns[i] = 1.0/winsq;

 Unknowns[Qwin+0] = 0;

 Unknowns[Qwin+1] = a0;
 Unknowns[Qwin+2] = a1;
 Unknowns[Qwin+3] = a2;
 Unknowns[Qwin+4] = b0;
 Unknowns[Qwin+5] = b1;
 Unknowns[Qwin+6] = b2;
 rl = 0;
 cl = 0;
 RefVar1 = 0.0;

}

//---------------------------------------------------------------------------
////Declare and Initialize coefficient matrices for LS Matching with
//four-quarter symmetry assumption
void Image::SetAffineLSResampSymMatchInitialValues()
{
 UnknownsNo = Qwin*Qwin+7;
 OldVar = RefVar = MAXDOUBLE;
 iterations = 0;
 CondNo = ResampHeight*ResampWidth;//no of equations
 TempHeight = TempImg->ImgHeight;
 TempWidth = TempImg->ImgWidth;

 try
 {
  p = new double[CondNo];
  corr = new double[UnknownsNo];
  U = new double[UnknownsNo];
  Unknowns = new double[UnknownsNo];
  M = new double*[UnknownsNo];
  for (int i = 0; i < UnknownsNo; i++)
   M[i] = new double[UnknownsNo];
  IM = new double*[TempHeight];
  for (int i = 0; i < TempHeight; i++)
   IM[i] = new double[TempWidth];
  winUnknowns = new double*[win];
  for(int i = 0; i < win; i++)
   winUnknowns[i] = new double[win];
  MM = new double*[UnknownsNo];
  for (int i = 0; i < UnknownsNo; i++)
   MM[i] = new double[UnknownsNo];
  for(i = 0; i< UnknownsNo; i++)
    {
     U[i] = 0;
     corr[i]=0;
     for(j=0;j<UnknownsNo;j++)
      M[i][j]=0;
    }
  for (i=0; i < TempHeight; i++)
   for (j = 0; j<TempWidth; j++)
     IM[i][j] = (double)TempImg->ImageMatrix[i][j];
  N = ResampHeight*ResampWidth*win*win; //No of Obs
  f = new double[CondNo];
  I = new double[CondNo];
  v = new double[N];
  B = new double*[CondNo];
  for (int i=0; i < CondNo; i++)
   B[i] = new double[UnknownsNo];
 }
 catch(...)
 {
  Application->MessageBox("Error Initializing Memory","Initialization failed", MB_OKCANCEL);
 }

 for (i = 0; i <win;i++)
  for(j=0;j<win; j++)
   winUnknowns[i][j] = 0.0;
 int winsq = win*win;
 for (i =0; i < Qwin*Qwin; i++)
  Unknowns[i] = 1.0/winsq;

 Unknowns[Qwin*Qwin+0] = 0;
 Unknowns[Qwin*Qwin+1] = a0;
 Unknowns[Qwin*Qwin+2] = a1;
 Unknowns[Qwin*Qwin+3] = a2;
 Unknowns[Qwin*Qwin+4] = b0;
 Unknowns[Qwin*Qwin+5] = b1;
 Unknowns[Qwin*Qwin+6] = b2;
 rl = 0;
 cl = 0;
 RefVar1 = 0.0;

}


//-------------------------------------------------------------------
//Declare and Initialize coefficient matrices for traditiona LS Matching
void Image::SetAffineLSInitialValues()
{
 UnknownsNo = 8;
 OldVar = RefVar = MAXDOUBLE;
 iterations = 0;
 CondNo =  TempHeight*TempWidth;
 try
 {
  p = new double[UnknownsNo];
  corr = new double[UnknownsNo];
  U = new double[UnknownsNo];
  Unknowns = new double[UnknownsNo];
  MM = new double*[UnknownsNo];
  Temp = new double*[UnknownsNo];
  M = new double*[UnknownsNo];
  for (int i = 0; i < UnknownsNo; i++)
   {
    M[i] = new double[UnknownsNo];
    MM[i] = new double[UnknownsNo];
    Temp[i] = new double[UnknownsNo];
   }
  for(i = 0; i< UnknownsNo; i++)
    {
     p[i]=0;
     corr[i]=0;
     U[i]=0;
     for(j=0;j<UnknownsNo;j++)
       M[i][j]=0;
    }
   TempHeight = TempImg->ImgHeight;
   TempWidth = TempImg->ImgWidth;
  N = TempHeight*TempWidth;
  f = new double[TempHeight*TempWidth];
  v = new double[TempHeight*TempWidth];
  PixMap = new int[TempHeight*TempWidth];
  Q = new double[TempHeight*TempWidth];
  for (int i = 0; i < TempHeight*TempWidth; i++)
  {
   Q[i] = 1;
   PixMap[i] = 0;
  }
  B = new double*[TempHeight*TempWidth];
  for (int i=0; i < TempHeight*TempWidth; i++)
   B[i] = new double[UnknownsNo];
 }
 catch(...)
 {
  Application->MessageBox("Error Initializing Memory","Initialization failed", MB_OKCANCEL);
 }

 Unknowns[0] = 0;
 Unknowns[1] = 1;
 Unknowns[2] = a0;
 Unknowns[3] = a1;
 Unknowns[4] = a2;
 Unknowns[5] = b0;
 Unknowns[6] = b1;
 Unknowns[7] = b2;
 rl = 1;
 cl = 1;
 RefVar1 = 0.0;
}

//-------------------------------------------------------------
//Start Executing LS matching with circular symmetry assumption
void Image::LSAffineResampCircularMatch()
{
 unsigned char DNL, DNR, DNU, DND, DN;
 double Rx,Ry, dn;
 float dist, refDist = sqrt(2);
 int rev_Uind, Uind,  Qwini, Qwinj;
 double l1, l2, fraction;

 double Maxcorr;
 // start least squares iterations
 while (iterations < Maxiterations && Maxcorr < iterationsThreshold)
  {
  // Get iteration time
  gettime(&t);
  Start = t.ti_hour*3600 +  t.ti_min * 60 + t.ti_sec + (float)t.ti_hund/100;
  int  Tempxl, Tempyl;
  ind = 0;
      h0 = Unknowns[cQwin+0];
      a0 = Unknowns[cQwin+1];
      a1 = Unknowns[cQwin+2];
      a2 = Unknowns[cQwin+3];
      b0 = Unknowns[cQwin+4];
      b1 = Unknowns[cQwin+5];
      b2 = Unknowns[cQwin+6];

  // loop through resampled image
  for (c=0; c <  ResampWidth; c++)
    for (r=0; r< ResampHeight; r++)
     {

       Tempxl = (int)(rl + r*JumpH +JumpH/2);
       Tempyl = (int)(cl + c*JumpW +JumpW/2);
       xl = rl + r;
       yl = cl + c;
       xr = a0 + a1*xl + a2*yl;
       yr = b0 + b1*xl + b2*yl;
```

```
DNL = BilinearResamp(xr,yr-1);
DNR = BilinearResamp(xr,yr+1);
DNU = BilinearResamp(xr-1,yr);
DND = BilinearResamp(xr+1,yr);
DN =  BilinearResamp(xr,yr);

Rx = (DND-DNU)/(2);
Ry = (DNR-DNL)/(2);

 double PixDN;
 dn = 0.0;
// GetQwinUnknownsArray();
for(i = 0; i <UnknownsNo; i++)
 B[ind][i] = 0.0;

 int UnknownsInd = 0;
 for(i=0; i< Qwin; i++)
  for(j =0; j< Qwin; j++)
   {
    Qwini = Qwin - i-1;
    Qwinj = Qwin - j -1;
    dist = sqrt(Qwini*Qwini+Qwinj*Qwinj);
    if (dist <= (Qwin -1)+.002)
     {
     if (i != win/2 && j != win/2)
       PixDN =  IM[Tempxl+i-win/2][Tempyl+j-win/2]
           +IM[Tempxl+i-win/2][Tempyl-j+win/2]
           +IM[Tempxl-i+win/2][Tempyl+j-win/2]
           +IM[Tempxl-i+win/2][Tempyl-j+win/2];
     else if (j == win/2 && i != win/2)
      PixDN =  IM[Tempxl+i-win/2][Tempyl]
           +IM[Tempxl-i+win/2][Tempyl];
     else if (i== win/2 && j !=win/2)
      PixDN =  IM[Tempxl][Tempyl+j-win/2]
           +IM[Tempxl][Tempyl-j+win/2];
     else
      PixDN = IM[Tempxl][Tempyl];

     Uind = dist / (refDist + .002) ;
     rev_Uind =  (cQwin - 1) - (Uind + 1);
     l1 = Unknowns[rev_Uind];
     l2 = Unknowns[rev_Uind + 1];
     fraction = (dist - Uind*refDist)/refDist;
     QwinUnknownsArray[UnknownsInd] = l2+ (l1-l2)* fraction;



     dn = dn + PixDN* QwinUnknownsArray[UnknownsInd];
     B[ind][rev_Uind] = B[ind][rev_Uind] + fraction * PixDN;
     B[ind][rev_Uind+1] = B[ind][rev_Uind+1] + (1-fraction) * PixDN;
     UnknownsInd++;
     }
    else
     {
     QwinUnknownsArray[UnknownsInd] = 0;
     UnknownsInd++;
     }

  }
 B[ind][cQwin+0] = -1;
 B[ind][cQwin+1] = -Rx;
 B[ind][cQwin+2] = -Rx*xl;
 B[ind][cQwin+3] = -Rx*yl;
 B[ind][cQwin+4] = -Ry;
 B[ind][cQwin+5] = -Ry*xl;
 B[ind][cQwin+6] = -Ry*yl;
 f[ind] = (h0+DN)-dn;// resampling, bilinear
```

```
        ind++;
      }

      //create window unknowns matrix
      int UnknownsInd = 0;
      for (i=0; i < Qwin; i++)
       for(j=0; j < Qwin; j++)
        {
         winUnknowns[i][j] = QwinUnknownsArray[UnknownsInd];
         winUnknowns[i][win-j-1] = QwinUnknownsArray[UnknownsInd];
         winUnknowns[win-i-1][j] = QwinUnknownsArray[UnknownsInd];
         winUnknowns[win-i-1][win-j-1] = QwinUnknownsArray[UnknownsInd];
         UnknownsInd++;
        }

      // Calculate W = A * AT (take advantage of symmetry)
       double sum = 0.0;
       for(int m = 0; m < win; m++)
         for(int n = 0; n <win; n++)
          sum = sum + winUnknowns[m][n]*winUnknowns[m][n];
       WMatrix = sum;

      //Compute M = BT *WL * B
       for (i = 0; i < UnknownsNo; i++)
         for (j = 0; j < UnknownsNo; j++)
          {
           item = 0;
           for (k = 0; k< CondNo; k++)
             item = item + B[k][i] * B[k][j];
           M[i][j] = item;
          }
      //comput U = BT *WL * f
       for(i = 0; i < UnknownsNo; i++)
        {
         double sum = 0;
         for(j = 0; j < CondNo; j++)
           sum = sum + B[j][i]*f[j];
           U[i] = sum ;
        }
     // Solve Normal Equations System
     choldc();
     cholsl();
     Maxcorr = GetMaxCorr(cQwin);

     AdjustSymObs();
     OldVar = RefVar;
     AdjustUnknowns();
     WriteLSResults();

      // Output Filter parameters
      fprintf(stream1, "%d %f \n", iterations, WMatrix);
      for (int i = 0; i < win; i++)
        {
         for (int j = 0; j < win; j++)
           fprintf(stream1, "%f ", winUnknowns[i][j]);
         fprintf(stream1, "\n");
        }
        iterations++;
  }

  GetUnknownsVCMtrix();
  FreeMemory();
}

//-------------------------------------------------------------------
//Start Executing LS matching with Diagonal symmetry assumption
void Image::LSAffineResampDiagMatch()
{
 Image *ResampImg = new Image(ResampWidth, ResampHeight);
 unsigned char DNL, DNR, DNU, DND, DN;
```

```
double Rx,Ry, dn;
float dist, refDist = sqrt(2);

int rev_Uind, Uind,  Qwini, Qwinj;
double l1, l2, fraction;

double Maxcorr;
while (iterations < Maxiterations && Maxcorr < iterationsThreshold)
 {
 gettime(&t);
 Start = t.ti_hour*3600 +  t.ti_min * 60 + t.ti_sec + (float)t.ti_hund/100;
 int  Tempxl, Tempyl;
 ind = 0;
      h0 = Unknowns[Qwin+0];
      a0 = Unknowns[Qwin+1];
      a1 = Unknowns[Qwin+2];
      a2 = Unknowns[Qwin+3];
      b0 = Unknowns[Qwin+4];
      b1 = Unknowns[Qwin+5];
      b2 = Unknowns[Qwin+6];

 for (c=0; c <  ResampWidth; c++)
    for (r=0; r< ResampHeight; r++)
     {
      Tempxl = (int)(rl + r*JumpH +JumpH/2);
      Tempyl = (int)(cl + c*JumpW +JumpW/2);
      xl = rl + r;
      yl = cl + c;
      xr = a0 + a1*xl + a2*yl;
      yr = b0 + b1*xl + b2*yl;
      DNL = BilinearResamp(xr,yr-1);
      DNR = BilinearResamp(xr,yr+1);
      DNU = BilinearResamp(xr-1,yr);
      DND = BilinearResamp(xr+1,yr);
      DN =  BilinearResamp(xr,yr);

      Rx = (DND-DNU)/(2);
      Ry = (DNR-DNL)/(2);

       double PixDN;
       dn = 0.0;
      // GetQwinUnknownsArray();
      for(i = 0; i <UnknownsNo; i++)
       B[ind][i] = 0.0;

       int x1,x2,y1,y2;
       int UnknownsInd = 0;
       for(i=0; i< Qwin; i++)
        for(j =0; j< Qwin; j++)
          {

          if (i != win/2 && j != win/2)
             PixDN =  IM[Tempxl+i-win/2][Tempyl+j-win/2]
                    +IM[Tempxl+i-win/2][Tempyl-j+win/2]
                    +IM[Tempxl-i+win/2][Tempyl+j-win/2]
                    +IM[Tempxl-i+win/2][Tempyl-j+win/2];
          else if (j == win/2 && i != win/2)
             PixDN =  IM[Tempxl+i-win/2][Tempyl]
                    +IM[Tempxl-i+win/2][Tempyl];
          else if (i== win/2 && j !=win/2)
             PixDN =  IM[Tempxl][Tempyl+j-win/2]
                    +IM[Tempxl][Tempyl-j+win/2];
          else
             PixDN = IM[Tempxl][Tempyl];

          Qwini = Qwin - i-1;
          Qwinj = Qwin - j -1;
          dist = sqrt(Qwini*Qwini+Qwinj*Qwinj);
          Uind = dist / (refDist + .02) ;
          rev_Uind =  (Qwin - 1) - (Uind + 1);
          l1 = Unknowns[rev_Uind];
          l2 = Unknowns[rev_Uind + 1];
          fraction = (dist - Uind*refDist)/refDist;
          QwinUnknownsArray[UnknownsInd] = l2+ (l1-l2)* fraction;

           dn = dn + PixDN* QwinUnknownsArray[UnknownsInd];
           B[ind][rev_Uind] = B[ind][rev_Uind] + fraction * PixDN;
           B[ind][rev_Uind+1] = B[ind][rev_Uind+1] + (1-fraction) * PixDN;
           UnknownsInd++;
          }

        B[ind][Qwin+0] = -1;
        B[ind][Qwin+1] = -Rx;
        B[ind][Qwin+2] = -Rx*xl;
        B[ind][Qwin+3] = -Rx*yl;
        B[ind][Qwin+4] = -Ry;
        B[ind][Qwin+5] = -Ry*xl;
        B[ind][Qwin+6] = -Ry*yl;

        f[ind] = (h0+DN)-dn;// resampling, bilinear

       //to save resampled image;
       ResampImg->ImageMatrix[xl][yl] = dn-h0;
        ind++;
      }

      //create window unknowns matrix
      int UnknownsInd = 0;
      for (i=0; i < Qwin; i++)
        for(j=0; j < Qwin; j++)
         {
          winUnknowns[i][j] = QwinUnknownsArray[UnknownsInd];
          winUnknowns[i][win-j-1] = QwinUnknownsArray[UnknownsInd];
          winUnknowns[win-i-1][j] = QwinUnknownsArray[UnknownsInd];
          winUnknowns[win-i-1][win-j-1] = QwinUnknownsArray[UnknownsInd];
          UnknownsInd++;
         }

       // Calculate W = A * AT (take advantage of symmetry)
         double sum = 0.0;
         for(int m = 0; m < win; m++)
           for(int n = 0; n <win; n++)
            sum = sum + winUnknowns[m][n]*winUnknowns[m][n];
         WMatrix = sum;

      //Compute M = BT *WL * B
       for (i = 0; i < UnknownsNo; i++)
          for (j = 0; j < UnknownsNo; j++)
            {
             item = 0;
             for (k = 0; k< CondNo; k++)
               item = item + B[k][i] * B[k][j];
             M[i][j] = item;
            }
      //comput U = BT *WL * f
       for(i = 0; i < UnknownsNo; i++)
        {
         double sum = 0;
         for(j = 0; j < CondNo; j++)
           sum = sum + B[j][i]*f[j];
//         U[i] = sum/WMatrix;
           U[i] = sum ;
        }

     choldc();
     cholsl();
     Maxcorr = GetMaxCorr(Qwin);
     AdjustSymObs();
     OldVar = RefVar;
     AdjustUnknowns();
```

```
    WriteLSResults();


     fprintf(stream1, "%d %f \n", iterations, WMatrix);
     for (int i = 0; i < win; i++)
       {
        for (int j = 0; j < win; j++)
          fprintf(stream1, "%f ", winUnknowns[i][j]);
        fprintf(stream1, "\n");
       }
      iterations++;
  }

  GetUnknownsVCMtrix();
 FreeMemory();
}


//---------------------------------------------------------------------
//Start Executing LS matching with four-quarter symmetry assumption
void Image::LSAffineResampSymMatch()
{
 Image *ResampImg = new Image(ResampWidth, ResampHeight);
 unsigned char DNL, DNR, DNU, DND, DN;
 double Rx,Ry, dn;
 double Maxcorr;
 while (iterations < Maxiterations && Maxcorr < iterationsThreshold)
  {
   gettime(&t);
   Start = t.ti_hour*3600 +  t.ti_min * 60 + t.ti_sec + (float)t.ti_hund/100;
   int  Tempxl, Tempyl;
   ind = 0;
       h0 = Unknowns[Qwin*Qwin+0];
       a0 = Unknowns[Qwin*Qwin+1];
       a1 = Unknowns[Qwin*Qwin+2];
       a2 = Unknowns[Qwin*Qwin+3];
       b0 = Unknowns[Qwin*Qwin+4];
       b1 = Unknowns[Qwin*Qwin+5];
       b2 = Unknowns[Qwin*Qwin+6];

   for (c=0; c <  ResampWidth; c++)// was r < TempHeight + 1
     for (r=0; r< ResampHeight; r++)// was c< TempWidth + 1
      {
       Tempxl = (int)(rl + r*JumpH +JumpH/2);
       Tempyl = (int)(cl + c*JumpW +JumpW/2);
       xl = rl + r;
       yl = cl + c;
       xr = a0 + a1*xl + a2*yl;
       yr = b0 + b1*xl + b2*yl;

       DNL = BilinearResamp(xr,yr-1);
       DNR = BilinearResamp(xr,yr+1);
       DNU = BilinearResamp(xr-1,yr);
       DND = BilinearResamp(xr+1,yr);
       DN =  BilinearResamp(xr,yr);

       Rx = (DND-DNU)/(2);
       Ry = (DNR-DNL)/(2);

        double PixDN;
        dn = 0.0;
        int UnknownsInd = 0;
        for(i=0; i< Qwin; i++)
          for(j =0; j< Qwin; j++)
           {
            if (i != win/2 && j != win/2)
              PixDN =  IM[Tempxl+i-win/2][Tempyl+j-win/2]
                    +IM[Tempxl+i-win/2][Tempyl-j+win/2]
                    +IM[Tempxl-i+win/2][Tempyl+j-win/2]
                    +IM[Tempxl-i+win/2][Tempyl-j+win/2];
        else if (j == win/2 && i != win/2)
          PixDN =  IM[Tempxl+i-win/2][Tempyl]
                +IM[Tempxl-i+win/2][Tempyl];
        else if (i== win/2 && j !=win/2)
          PixDN =  IM[Tempxl][Tempyl+j-win/2]
                +IM[Tempxl][Tempyl-j+win/2];
        else
          PixDN = IM[Tempxl][Tempyl];

        dn = dn + PixDN*Unknowns[UnknownsInd];
        B[ind][UnknownsInd] = PixDN;
        UnknownsInd++;
       }

     B[ind][Qwin*Qwin+0] = -1;
     B[ind][Qwin*Qwin+1] = -Rx;
     B[ind][Qwin*Qwin+2] = -Rx*xl;
     B[ind][Qwin*Qwin+3] = -Rx*yl;
     B[ind][Qwin*Qwin+4] = -Ry;
     B[ind][Qwin*Qwin+5] = -Ry*xl;
     B[ind][Qwin*Qwin+6] = -Ry*yl;


     f[ind] = (h0+DN)-dn;// resampling, bilinear

    //To save resampled image
    ResampImg->ImageMatrix[xl][yl] = dn-h0;
     ind++;
  }
  //create window unknowns matrix
  int UnknownsInd = 0;
  for (i=0; i < Qwin; i++)
    for(j=0; j < Qwin; j++)
     {
      winUnknowns[i][j] = Unknowns[UnknownsInd];
      winUnknowns[i][win-j-1] = Unknowns[UnknownsInd];
      winUnknowns[win-i-1][j] = Unknowns[UnknownsInd];
      winUnknowns[win-i-1][win-j-1] = Unknowns[UnknownsInd];
      UnknownsInd++;
     }

   // Calculate W = A * AT (take advantage of symmetry)
    double sum = 0.0;
    for(int m = 0; m < win; m++)
      for(int n = 0; n <win; n++)
       sum = sum + winUnknowns[m][n]*winUnknowns[m][n];
    WMatrix = sum;

  //Compute M = BT *WL * B
   for (i = 0; i < UnknownsNo; i++)
     for (j = 0; j < UnknownsNo; j++)
       {
        item = 0;
        for (k = 0; k< CondNo; k++)
          item = item + B[k][i] * B[k][j];
        M[i][j] = item;
       }
  //comput U = BT *WL * f
   for(i = 0; i < UnknownsNo; i++)
    {
     double sum = 0;
     for(j = 0; j < CondNo; j++)
       sum = sum + B[j][i]*f[j];
       U[i] = sum ;
    }

choldc();
cholsl();
Maxcorr = GetMaxCorr(Qwin * Qwin);
```

```
    AdjustSymObs();
    OldVar = RefVar;
    AdjustUnknowns();
    WriteLSResults();


     fprintf(stream1, "%d %f \n", iterations, WMatrix);
     for (int i = 0; i < win; i++)
      {
       for (int j = 0; j < win; j++)
         fprintf(stream1, "%f ", winUnknowns[i][j]);
       fprintf(stream1, "\n");
      }

      iterations++;
  }

  GetUnknownsVCMtrix();
  FreeMemory();
}
//------------------------------------------------------------------------
//Start Executing Traditiona LS matching
void Image::LSAffineMatch()
{

 unsigned char DNL, DNR, DNU, DND, DN;
 double Rx,Ry,**fmatrix;
 double Maxcorr;
 fmatrix = new double*[TempHeight];
   for (int i=0; i< TempHeight; i++)
    fmatrix[i] = new double[TempWidth];
  iterations = 0;

  while (iterations < Maxiterations && Maxcorr < iterationsThreshold)
  {
    ind = 0;
    gettime(&t);
    Start = t.ti_hour*3600 +  t.ti_min * 60 + t.ti_sec + (float)t.ti_hund/100;
    int ObsCount = 0;
    WMatrix = 1.0;
    for (r=0; r <  TempHeight; r++)
      for (c=0; c< TempWidth; c++)
       {
        h0 = Unknowns[0];
        h1 = Unknowns[1];
        a0 = Unknowns[2];
        a1 = Unknowns[3];
        a2 = Unknowns[4];
        b0 = Unknowns[5];
        b1 = Unknowns[6];
        b2 = Unknowns[7];

        xl = rl + r -1;
        yl = cl + c -1;
        xr = a0 + a1*xl + a2*yl;
        yr = b0 + b1*xl + b2*yl;

        //Resampling needed;
        //using Bilinear

        DNL = BilinearResamp(xr,yr-1);
        DNR = BilinearResamp(xr,yr+1);
        DNU = BilinearResamp(xr-1,yr);
        DND = BilinearResamp(xr+1,yr);
        DN =  BilinearResamp(xr,yr);

        Rx = (DND-DNU)/2;
        Ry = (DNR-DNL)/2;
        if (PixMap[ObsCount] == 0)
        { //start if
       B[ind][0] = 1;
       B[ind][1] = DN;// resampling, Bilinear
       B[ind][2] = Rx*h1;
       B[ind][3] = Rx*h1*xl;
       B[ind][4] = Rx*h1*yl;
       B[ind][5] = Ry*h1;
       B[ind][6] = Ry*h1*xl;
       B[ind][7] = Ry*h1*yl;

       f[ind] = TempImg->ImageMatrix[xl][yl]-(h0+h1*DN);// resampling, bilinear

       fmatrix[r][c] =  f[ind];
       ind++;
      }
      ObsCount++;
     }

    for (i = 0; i < UnknownsNo; i++)
     for (k = 0; k < UnknownsNo; k++)
       if (k >= i)    //Upper triangle
        {
         item = 0;
         for (j = 0; j< N; j++)
           item = item + B[j][i]*Q[j] * B[j][k];
         M[i][k] = item;

        }


    // Calculate U = BT * f
   for (i = 0; i < UnknownsNo; i++)
     {
      item = 0;
      for (j = 0; j< N; j++)
        item = item + B[j][i]*Q[j] * f[j];
      U[i] = item;
     }

   //get (BT*B)^-1
   choldc();
   cholsl();
   Maxcorr = GetMaxCorr(1);
   OldVar = RefVar;
   GetVariance();
   AdjustUnknowns();

   WriteLSResults();
   iterations++;

  } //end while
 GetUnknownsVCMtrix();
 FreeMemory();
}

//--------------------------------------------------------------------------------
//Perform resampling using Bilinear Interpolation method
unsigned char Image::BilinearResamp( float x, float y)
{
 long xr, yc;
 unsigned char dn;
 float dx, dy, dn1, dn2;
 xr = (int)x;
 yc = (int)y;
 dx = x - xr;
 dy = y - yc;

 dn1 = ImageMatrix[xr][yc]+ dx * (ImageMatrix[xr+1][yc]-ImageMatrix[xr][yc]);
 dn2 = ImageMatrix[xr][yc+1] + dx*(ImageMatrix[xr+1][yc+1]-ImageMatrix[xr][yc+1]);
 dn = (unsigned char)(dn1 + dy * (dn2 - dn1));
 return dn;
```

```
}
//------------------------------------------------------------------------
//Compute Observation residuals (case of resampling within LS matching)
void Image::AdjustSymObs()
{
 double sum;
 long ind;

 //calculate  I =f - B*corr
 for (i = 0; i < CondNo; i++)
  {
   sum = 0;
   for (j =0; j <UnknownsNo; j++)
     sum = sum + B[i][j]*corr[j];
   I[i] = f[i]-sum;

  }


// calculate v

   ind = 0;
   for (int i = 0 ; i < ResampWidth*ResampHeight; i++)
        for (int m = 0 ; m < win; m++)
          for(int n = 0; n < win; n++)
             v[ind++] = (winUnknowns[m][n]*I[i])/ WMatrix;

    RefVar = 0.0;
    for (i = 0; i < N; i++)
       RefVar = RefVar + v[i]*v[i];

   RefVar = RefVar / (CondNo-UnknownsNo);


}
//------------------------------------------------------------------------
//Compute Observation residuals (case of traditional LS matching)
void Image::AdjustObs()
{
 double sum;
 RefVar = 0;


 //calculate  I =f - B*corr
 for (i = 0; i < CondNo; i++)
  {
   sum = 0;
   for (j =0; j <UnknownsNo; j++)
     sum = sum + B[i][j]*corr[j];
   I[i] = f[i]-sum;
  }


   int ind = 0;
   for (int j = 0 ; j < ResampHeight;j++)
    for (int m = 0 ; m < win; m++)
     for (int i = 0 ; i < ResampWidth; i++)
      for(int n = 0; n < win; n++)
       {
        v[ind] = (Unknowns[m*win+n]*I[i*ResampHeight+j])/ WMatrix;
        ind++;
       }


    for (i = 0; i < N; i++)
       RefVar = RefVar + v[i]*v[i];

   RefVar = RefVar / (CondNo-UnknownsNo);
}
//------------------------------------------------------------------------
```

```
//Inverse square matrix using Chelosky's method, code modified from (Press et al.1988).
void Image::MatrixInverse(double **W,int size)
{
 int i,j,k;
 double sum;
 int CondNo = size;
 for (i=0; i<CondNo; i++) {
  for (j=i; j<CondNo; j++) {
   for (sum = W[i][j], k = i-1; k >=0; k--) sum -= W[i][k]*W[j][k];
   if (i==j) {
    if (sum <= 0.0)
      Application->MessageBox("Not Positive Definite Matrix","choldc failed", MB_OKCANCEL);
    p[i] = sqrt(sum);
   } else W[j][i] = sum/p[i];
  }
 }

 for (i=0; i < CondNo; i++) {
  W[i][i] = 1.0/p[i];
  for (j=i+1; j<CondNo; j++) {
   sum = 0.0;
   for(k=i; k < j ; k++) sum -= W[j][k]*W[k][i];
    W[j][i] = sum/p[j];
  }
 }
}

//------------------------------------------------------------------------------
//Soloving linear algebraic equations, code modified from (Press et al.1988).
void Image::choldc()
{
 int i,j,k;
 double sum;

 for (i=0; i<UnknownsNo; i++) {
  for (j=i; j<UnknownsNo; j++) {
   for (sum = M[i][j], k = i-1; k >=0; k--) sum -= M[i][k]*M[j][k];
   if (i==j) {
    if (sum <= 0.0)
      Application->MessageBox("Not Positive Definite Matrix","choldc failed", MB_OKCANCEL);
    p[i] = sqrt(sum);
   } else M[j][i] = sum/p[i];
  }
 }
}


void Image::cholsl()
{
 int i,k;
 double sum;
 for (i=0; i<UnknownsNo; i++) {
   for (sum = U[i], k = i-1; k >= 0; k--) sum -= M[i][k]*corr[k];
   corr[i] = sum / p[i];
 }

 for (i=UnknownsNo - 1; i>=0; i--) {
   for (sum = corr[i], k = i+1; k<=UnknownsNo-1; k++) sum -=M[k][i]*corr[k];
    corr[i] = sum/p[i];
 }
}
//------------------------------------------------------------------------------

//Compute Adjusted Unknowns
void Image::AdjustUnknowns()
{
 for (i = 0 ; i< UnknownsNo; i++)
  Unknowns[i] = Unknowns[i] + corr[i];
}
```

```
//-----------------------------------------------------------------------------
// Output Affine parameters and iteration time for each iteration
void Image::WriteLSResults()
{
 gettime(&t);
 End = t.ti_hour*3600 +  t.ti_min * 60 + t.ti_sec + (float) t.ti_hund/100;
// fprintf(stream, "%d %f %f %f %f %f %f %f %f %f %f %f %f %f\n", iterations, a0+rul, a1,
//               a2,b0,b1,b2, c1, c2, h0, h1, (double)RefVar, (double)RefVar1, (End-Start));
 fprintf(stream, "%d %f %f %f  %f %f %f %f %f  %f\n", iterations, a0+rul, a1,
              a2,b0,b1,b2, h0, h1, (End-Start));

}

//-----------------------------------------------------------
//Compute V-C matrix for unknowns
void Image::GetUnknownsVCMtrix()
{
 int i,j=0;

 MatrixInverse(M, UnknownsNo);
 for(i=0; i< UnknownsNo; i++)
  for(j =0 ;j < UnknownsNo ;j++)
   if (j > i)
     M[i][j] = 0;

 for (i = 0; i < UnknownsNo; i++)
  for(j = 0 ; j < UnknownsNo; j++)
        {
         item = 0;
         for(k = 0; k< UnknownsNo; k++)
           item = item + M[k][i] * M[k][j];
         MM[i][j] = item;
        }



 fprintf(stream, "\n Varianc-Covariance for Unknowns \n");

 for(i=0; i< UnknownsNo; i++)
    fprintf(stream, "%f %f \n",Unknowns[i], MM[i][i]*WMatrix);

}

//-----------------------------------------------------------------------------
// Perform Correlation Matching (case of resampling within matching)
void Image::PerformResampCorrMatching(Image *TImg, int ResampWinSize, int ResampH,
                          int ResampW, int rtol, int ctol)
{
 int Diff = 0, DN;
 double Corrf = 0.0, amax, bmax;
 CorrMax = 0;
 float Jump,dn;
 int Tempxl, Tempyl;
 TempImg = TImg;
 win = ResampWinSize;
 ResampHeight = ResampH;
 ResampWidth = ResampW;
 Image *ResampImg = new Image(ResampWidth, ResampHeight);
 JumpW =  (float)TempImg->ImgWidth / ResampWidth;
 JumpH = (float)TempImg->ImgHeight / ResampHeight;
 if (JumpH < JumpW)
   Jump = JumpH;
 else Jump = JumpW;
 if (win >= Jump)
    win = (int)(Jump-1);

 for (c=0; c <  ResampWidth; c++)
    for (r=0; r< ResampHeight; r++)
      {
```

```
      Tempxl = (int)(r*JumpH +JumpH/2);
      Tempyl = (int)(c*JumpW +JumpW/2);
      dn = 0.0;
      for(i=0; i< win; i++)
       for(j =0; j< win; j++)
           dn = dn + TempImg->ImageMatrix[Tempxl+i-win/2][Tempyl+j-win/2];
      ResampImg->ImageMatrix[r][c] = dn/(win*win);
     }

 TempWidth = ResampWidth;
 TempHeight = ResampHeight;
 N = TempWidth * TempHeight;
 r0 = a0 - rtol; //for now
 c0 = b0 - ctol;  //for now
 rMax = a0+ TempHeight + rtol;
 cMax = b0 + TempWidth + ctol;
 rowsNo = rMax - r0;
 colsNo = cMax - c0;

 for (i = 0; i < rowsNo - TempHeight; i++)
  for (j = 0; j < colsNo - TempWidth; j++)
   {

    SigmaX = SigmaY = SigmaXY = SigmaX2 = SigmaY2 = 0;

    for (r = 0; r <TempHeight; r++)
      for (c = 0; c <TempWidth; c++)
      {

       xl = r ;
       yl = c;
       xr = r0 + i + a1*xl + a2*yl;
       yr = c0 + j + b1*xl + b2*yl;

       DN =  BilinearResamp(xr,yr);

       SigmaX = SigmaX + ResampImg->ImageMatrix[r][c];
       SigmaY = SigmaY + DN;
       SigmaXY= SigmaXY + ResampImg->ImageMatrix[r][c] * DN;
       SigmaX2 = SigmaX2 + ResampImg->ImageMatrix[r][c] * ResampImg->ImageMatrix[r][c];
       SigmaY2 = SigmaY2 + DN*DN;
       }
     SXY =  SigmaXY-SigmaX*SigmaY/N;
     SXX =  SigmaX2-SigmaX*SigmaX/N;
     SYY =  SigmaY2-SigmaY*SigmaY/N;

      if (SXX == 0 || SYY == 0)
       int g = 0;
     Corrf = fabs(SXY /sqrt(SXX*SYY));

      if (Corrf > CorrMax) {
       CorrMax = Corrf;
       amax = i;
       bmax = j;
      }
    }

     a0 = r0 + amax;
     b0 = c0 +  bmax;

    delete ResampImg;

}

//---------------------------------------------------------------------------
// Perform Correlation Matching (case of no resampling within matching)
void Image::PerformCorrMatching(Image *TImg,int rtol,int ctol)
{
 int Diff = 0, DN;

 double Corrf=0.0, amax, bmax;
 CorrMax = 0;
 TempImg = TImg;
 TempWidth = TempImg->ImgWidth;
 TempHeight = TempImg->ImgHeight;
 N = TempWidth * TempHeight;
 r0 = a0 - rtol;
 c0 = b0 - ctol;
 rMax = a0+ TempHeight + rtol;
 cMax = b0 + TempWidth + ctol;

 rowsNo = rMax - r0;
 colsNo = cMax - c0;

 for (i = 0; i < rowsNo - TempHeight; i++)
  for (j = 0; j < colsNo - TempWidth; j++)
   {


    SigmaX = SigmaY = SigmaXY = SigmaX2 = SigmaY2 = 0;
    for (r = 0; r <TempHeight; r++)
      for (c = 0; c <TempWidth; c++)
      {

       xl = r ;
       yl = c;
       xr = r0 + i + a1*xl + a2*yl;
       yr = c0 + j + b1*xl + b2*yl;

       DN =  BilinearResamp(xr,yr);
       SigmaX = SigmaX + TempImg->ImageMatrix[r][c];
       SigmaY = SigmaY + DN;
       SigmaXY= SigmaXY + TempImg->ImageMatrix[r][c] * DN;
       SigmaX2 = SigmaX2 + TempImg->ImageMatrix[r][c] * TempImg->ImageMatrix[r][c];
       SigmaY2 = SigmaY2 + DN*DN;
       }
     SXY =  SigmaXY-SigmaX*SigmaY/N;
     SXX =  SigmaX2-SigmaX*SigmaX/N;
     SYY =  SigmaY2-SigmaY*SigmaY/N;

     Corrf = fabs(SXY /sqrt(SXX*SYY));
      if (Corrf > CorrMax) {
       CorrMax = Corrf;
       amax = i;
       bmax = j;
      }

    }
     a0 = r0 + amax;
     b0 = c0 +  bmax;
}

//------------------------------------------------------
//Filtering Images with Gaussian Filter
void Image::PerformGaussianFilter(int Gwin, float sigma)
{
   float *Gausswin;
   double  GPixValue, WeightFactor = 0;
   Graphics::TBitmap* GImageBitmap;
   Gausswin = new float[Gwin];
   float ** TempGaussImageMatrix;
   unsigned char ** GaussImageMatrix;
   TempGaussImageMatrix = new float * [ImgHeight];
   for (int i=0; i < ImgHeight; i++)
    TempGaussImageMatrix[i] = new float [ImgWidth];

   GaussImageMatrix = new unsigned char * [ImgHeight];
```

```
for (int i=0; i < ImgHeight; i++)
  GaussImageMatrix[i] = new unsigned char [ImgWidth];

for (i = 0; i < Gwin; i++)
  Gausswin[i] =  exp(- (i - Gwin/2)* (i - Gwin/2)/2/sigma/sigma);

 for (i = 0; i < Gwin; i++)
      for (int j = 0; j < Gwin; j++)
        WeightFactor = WeightFactor + Gausswin[i]* Gausswin[j];


GPixValue = 0;
for (i=0;i<ImgHeight;i++)
  for (j=0;j<ImgWidth;j++)
  {
    GPixValue = 0;
    for (int m =0; m<Gwin; m++)
    {
     if (i >= Gwin/2 && i < (ImgHeight - Gwin/2))
       GPixValue = GPixValue + ImageMatrix[i+m -Gwin/2][j] * Gausswin[m];
     else if ((i < Gwin/2 && m < Gwin/2)|| (i >= (ImgHeight -Gwin/2) && m >Gwin/2))
       GPixValue = GPixValue + ImageMatrix[i-m +Gwin/2][j] * Gausswin[m];
     else
       GPixValue = GPixValue + ImageMatrix[i+m -Gwin/2][j] * Gausswin[m];
    }
    TempGaussImageMatrix[i][j] = GPixValue;
 }

 for (i=0;i<ImgHeight;i++)
  for (j=0;j<ImgWidth;j++)
  {
    GPixValue = 0;
    for (int m =0; m<Gwin; m++)
    {
     if (j >= Gwin/2 && j < (ImgHeight - Gwin/2))
       GPixValue = GPixValue + TempGaussImageMatrix[i][j+m -Gwin/2] * Gausswin[m];
     else if ((j < Gwin/2 && m < Gwin/2)|| (j >= (ImgHeight -Gwin/2) && m >Gwin/2))
       GPixValue = GPixValue + TempGaussImageMatrix[i][j-m +Gwin/2] * Gausswin[m];
     else
       GPixValue = GPixValue + TempGaussImageMatrix[i][j+m -Gwin/2] * Gausswin[m];
    }
    GaussImageMatrix[i][j] = (unsigned char)(GPixValue/WeightFactor);
 }

for (int i = 0; i < ImgHeight ; i++)
   delete[] TempGaussImageMatrix[i];
delete[] TempGaussImageMatrix;

GImageBitmap = new Graphics::TBitmap;
GImageBitmap->Width = ImgWidth;
GImageBitmap->Height = ImgHeight;


for (int i =0; i < ImgHeight; i++)
 for (int j = 0; j < ImgWidth; j++)
  {

   int pixel =   GaussImageMatrix[i][j];
   GImageBitmap->Canvas->Pixels[j][i] = RGB(pixel, pixel, pixel);
  }

for (int i = 0; i < ImgHeight ; i++)
   delete[] GaussImageMatrix[i];
 delete[] GaussImageMatrix;
TSaveDialog *SaveDialog = new TSaveDialog(NULL);
if (GImageBitmap)
 if (SaveDialog->Execute())
  {
   GImageBitmap->SaveToFile(SaveDialog->FileName);
   delete SaveDialog;
   }
 else
   Application->MessageBox("No BitMap Created","BitMap Save Error",MB_OKCANCEL);

}
//---------------------------------------------------------------
//Convert Window parameters unknowns from array to window format
void Image::GetQwinUnknownsArray()
{
   float dist, refDist = sqrt(2);
   int Uind; ind = 0;
   double l1, l2;
   for (int i = Qwin; i>0; i--)
      for (int j = Qwin; j>0; j--)
       {
        dist = sqrt(i*i+j*j);
        Uind = dist / refDist;
        l1 = Unknowns[Uind];
        l2 = Unknowns[Uind + 1];

        QwinUnknownsArray[ind++] = l1+ (l2-l1)* (dist - Uind*refDist)/refDist   ;

       }
}

 //-------------------------------------------------------------------------------
//Compute maximum correction to the 4 rotation affine transformation
//parameters (used to compare with threshold to stop iterations)
int Image::GetMaxCorr(int winParNo)
{
 double a1corr, a2corr, b1corr, b2corr, Maxcorr;

 a1corr = Unknowns[winParNo+2];
 a2corr = Unknowns[winParNo+3];
 b1corr = Unknowns[winParNo+5];
 b2corr = Unknowns[winParNo+6];
 if (a1corr <  a2corr) Maxcorr = a2corr;
 else if (a1corr <  b1corr) Maxcorr = b1corr;
 else if (a1corr <  b2corr) Maxcorr = b2corr;
 else Maxcorr = a1corr;

 return Maxcorr;

}
//-------------------------------------------------------------------------------
//close files open for output results
void Image::CloseOutputFiles()
{
  fclose(stream);
  fclose(stream1);

}

//-------------------------------------------------------------------------------
//Open files to output results
void Image::OpenOutputFiles(char * TempNme)
 {

   TempName = strcat(TempNme,".txt");
  if ((stream = fopen(TempName, "wt")) == NULL) /* open file TEST.$$$ */
     Application->MessageBox("File I/O error","Failure to Open Output File", MB_OKCANCEL);
  fprintf(stream, " Least Squares Results \n");

  //ASCII file for output filter parameters
  TempName = strcat(TempNme,".par");
  if ((stream1 = fopen(TempName, "wt")) == NULL) /* open file TEST.$$$ */
     Application->MessageBox("File I/O error","Failure to Open Output File", MB_OKCANCEL);
  fprintf(stream1, " Window Parameters: \n");

 }
```

```
//------------------------------------------------------------------------------
//Free memore reserved for LS coefficient and processing matrices
void Image::FreeMemory()
{
 delete[] p;
 delete[] corr;
 delete[] U;
 delete[] Unknowns;

 for (int i = 0; i < UnknownsNo; i++)
  {
   delete[] M[i];
   delete[] MM[i];
  }
 delete[] M;
 delete[] MM;
 delete[] f;
 delete[] v;
 for (int i = 0; i < CondNo ; i++)
    delete[] B[i];
 delete[] B;

}

//-----------------------------------------------------------------------
```

LIST OF REFERENCES

Abd-Elrahman, A., L. Pearlstine, S. Smith, and P. Princz, 2000, Use of Small Format Digital Aerial Images For Classification of Satellite Images, Proceedings of XVI IMEKO World Congress, IMEKO 2000 conference, September, 2000, Vienna, Austria. Vol. VII, pp. 273-278.

Abd-Elrahman. A., L. Pearlstine, B. Dewitt, and S. Smith, 2001. Detection of Positional Errors in Systems Utilizing Small Format Digital Aerial Imagery and Navigation Sensors Using Area-based Matching Techniques. Photogrammetric Engineering and Remote Sensing Vol. 67, No. 7, pp. 825-832.

Ackermann, F., 1983. High Precision Digital Image Correlation. In proceedings of 39th photogrammetric week. Institute of Photogrammetry, University of Stuttgart, pp. 231-243.

Ackermann, F. and H. Schade, 1993. Applications of GPS for Aerial Triangulation. . Photogrammetric Engineering and Remote Sensing, Vol. 59, No. 11, pp. 1625-1632.

Agouris P., 1992. Multiple Image Multipoint Matching for Automatic Orientation, PhD Dissertation, Dept. of Geodetic Science, Ohio State University, Ohio.

Baltsavias E.P., 1999. Airborne Laser Scanning: basic relations and formulas. ISPRS Journal of Photogrammetry Remote Sensing, Vol 54, pp. 199-214.

Barnea, D.I. and H.F. Silverman, 1972. A Class of Algorithms for Fast Digital Image Registration, IEEE Transactions on Computer, Vol. 21, No 2, pp. 179-186.

Bracewell, N. R., 1986. The Fourier Transform and its Applications. McGraw-Hill, New York.

Billingsley, F. C., P. W. Anuta, J.L. Carr, C.D. McGillem, D.M. Smith and T.C. Strand, 1983. Manual of Remote Sensing (2nd Ed.). American Society of Photogrammetry. Falls Church, Va.

Brown, L.G., 1992. A Survey of Image Registration Techniques. ACM Computing Surveys, Dec 1992, Vol. 24, No. 4.

Buhrke, M.K., 1991. A Comparison of USGS 7.5 Minutes Quadrangle Maps with Ground Surveying for the Rectification of SPOT Panchromatic Imagery. M.Sc. Thesis, University of Florida, Gainesville, FL.

Burke, M.W, 1996. Image Acquisition. Chapman & Hall, London, UK.

Calitz, M.F. and H. Ruther, 1996. Least Absolute Deviation (LAD) Image Matching. ISPRS Journal of Photogrammetry & Remote Sensing, Vol. 51, 1996, pp. 223-229.

Castleman, K. R., 1996. Digital Image Processing. Prentice-Hall, Inc, New Jersey.

Curry, S. and K. Schuckman, 1993. Practical Considerations for the Use of Airborne GPS for Photogrammetry. Photogrammetric Engineering and Remote Sensing, Vol. 59, No. 11, pp. 1611-1617.

Djamdji, J.P., A. Bijaoui, and R. Maniere, 1993. Geometrical Registration of Images: the multiresolution approach. Photogrammetric Engineering and Remote Sensing, Vol. 59, No. 5, pp. 645-653.

Fonseca, L. M.G. and B.S. Manjunath, 1996. Registration Techniques for Multisensor Remotely Sensed Imagery. Photogrammetric Engineering and Remote Sensing, Vol. 62, No. 9, pp. 1049-1056.

Goshtasby, A. 1986. Piecewise Linear Mapping Functions For Image Registration. Pattern Recognition, Vol. 19, No. 6, pp. 459-466.

Goshtasby, A, 1987. Piecewise Cubic Mapping Functions For Image Registration. Pattern Recognition, Vol. 20, No. 5, pp. 525-533.

Greenfield, J.S., 1991. An Operator Based Matching System. Photogrammetric Engineering and Remote Sensing Vol. 57, No. 8, pp. 1049-1055.

Gruen, A, 1985. Adaptive Least Squares Correlation – a powerful matching technique. South Africa Journal of Photogrammetric Engineering and Remote Sensing, Vol. 53, No.2, pp. 167-169.

Gruen, A., M. Cocard and H.G. Kahle, 1993. Photogrammetry and Kinematic GPS: Results of a high accuracy test. Photogrammetric Engineering and Remote Sensing, Vol. 59, No. 11, pp. 1643-1650.

Hein, G.W., 1989. Precise Kinematic GPS/INS Positioning: A discussion on the applications in aerophotogrammetry. Proceedings of 42nd photogrammetric week, Stuttgart, Germany, pp. 261-282.

Helava, U. V, 1988. Object-Space Least Squares Correlation. Photogrammetric Engineering & Remote Sensing, Vol. 54, No. 6, pp. 711-714.

Huang, Y. 1996, DTM Generation From Digitized Aerial Photos of A Complex Scene by Employing Pattern Recognition with the Fourier Transform and Multi Resolution Feature-Based Stereo Matching. Ph.D. dissertation, University of Florida, Gainesville, Fl.

Jacobsen, K., 1988. SPOT Image Evaluations for Cartographic Purposes. Proceedings of the international symposium on topographic applications of SPOT data, Sherbrooke, Quebec, Canada.

Keys, R.G., 1981. Cubic Convolution Interpolation for Digital Image Processing. IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP. 29(6), December, 1981, pp. 1153-1160.

Konency, G., P. Lohmann, H. Engel and E. Kruck, 1987. Evaluation of SPOT Imagery on Analytical Photogrammetric Instruments. Photogrammetric Engineering and Remote Sensing, Vol. 53, No. 9, pp1223-1230.

Krabil, W.B., 1989. GPS Applications to Laser Profiling and Laser Scanning for Digital Terrain Models. Proceedings of 42nd photogrammetric week, Stuttgart, Germany, pp. 261-282.

Kruce, F.A., 1988. Use of Airborne Imaging Spectrometer Data to Map Minerals Associated with Hydrothermally Altered Rocks in the Northern Grapevine Mountains, Nevada and California. Remote Sensing of Environment. Vol. 24, No. 1, pp. 31-51.

Lecki, D.G., 1980. Use of Polynomial Transformations for Registration of Airborne Digital Line Scan Images. 14th International Symposium on Remote Sensing of Environment. Pp. 635-641.

Lemmens, M., 1988. A Survey on Stereo Matching Techniques. International archives of Photogrammetry and Remote Sensing, 27 (B3), pp. 11-23.
Li, R.,1997. Mobile Mapping: An emerging technology for spatial data acquisition. . Photogrammetric Engineering and Remote Sensing. Vol. 63, No. 9, pp. 1085-1092.

Lillesand, T. M. and R.W. Kiefer, 1994. Remote Sensing and Image Interpretation. John Wiley & Sons, New York.

Lorre, J.J. and A.R. Gillespie, 1980. Artifacts in Digital Images. SPIE Applications of Digital Image Processing in Astronomy, Vol. 264, pp. 123-135.

Mikhail, E.M., 1976. Observations and Least Squares. Thomas Y. Crowell Company, Inc, New York.

Mitchell, D.P. and A. N. Netravali, 1988. Reconstruction Filters in Computer Graphics. Computer Graphics. Vol. 22, No. 2, pp. 221-228.

Moik, J.G., 1980. Digital Processing of Remotely Sensed Images. NASA SP-431. Washington, D.C., Government Printing Office.

Morris, A.C., A. Stenves, and J. –P. A. L. Muller, 1988. Ground Control Determination for Registration of Satellite Imagery using Digital Map Data. Photogrammetric record, pp. 809-822.

Novak, K., 1992. Rectification of Digital Imagery. Photogrammetric Engineering and Remote Sensing, Vol. 58, No. 3, pp 339-344.

Orndorff, E.M., 1997. Horizontal Accuracy Assessment of a USGS DOQQ Using a Differential GPS Methodology. Proceedings of GIS/LIS '97 Annual Conference and Exposition, Cincinnati, Ohio, pp. 814-817.

Pearlstine, L. and S. Smith, L. Ojanen, J. Stenberg, and A. Abd-Elrahman, 2000. Land Cover Classification and Mapping in North Florida. U.S.G.S.Biological Resources Division, Florida Cooperative Fish and Wildlife Research Unit. University of Florida, Gainesville, FL, Technical Report No. 63.

Press, W., B. Flannery, S. Teukolsky, and W. Vetterling, 1988. Numerical Recipes in C: The Art of Scientific Computing. Cambridge UP.

Rosenholm D., 1987. Multi-Point Matching Using the Least Squares Technique for Evaluation of Three-Dimensional Models. Photogrammetric Engineering and Remote Sensing, Vol. 53, No. 6, pp. 621-626.

Rossenfeld, A. and A.C. Kak 1982, Digital Picture Processing. Vol. I and II. Academic Press, Orlando, Fl.

Schenk, T., J. C. Li, and C. Toth, 1991. Towards an Automatic System for Orienting Digital Stereopairs. Photogrammetric Engineering and Remote Sensing, Vol. 57, No. 8, pp. 1057-1064.

Schowengerdt, R. A., 1997. Remote Sensing Models and Methods for Image Processing. Academic Press, New York.

Schwarz, K.-P., M.A. Chapman, M.E. Cannon, and P. Gong, 1993. An Integrated INS/GPS Approach to the Georeferencing of Remotely Sensed Data. Photogrammetric Engineering and Remote Sensing. Vol. 59, No. 11, pp. 1667-1674.

Schwarz, K.-P., and N. El-Sheimy, 1996. Kinematic Multi-sensor Systems for Close Range Digital Imaging. International Archives of Photogrammetry and Remote Sensing, 31(B3), pp. 774-785.

Smith, S.E., B.A. Dewitt, E.P. Gonzalez, and G.W. Hurt, 1995. Georeferencing Satellite Imagery of Digital Soil Mapping. Surveying and Land Information Systems, Vol. 55, No. 1, pp. 13-20.

Stefanidis, A., 1993. Using Scale Space Techniques to Estimate Scale difference across Images. PhD Dissertation, Department of Geodetic Science and surveying, Ohio State University, Ohio.

Stockman, G.C., S. Kopstein and S. Benett 1982. Matching Images to Models for Registration and Object Detection via Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 4, pp. 229-241.

Sun, B.L., 1992. Geometric Control of SPOT Panchromatic Imagery Using Digitized Aerial Photographs. Ph.D. Dissertation, University of Wisconsin, Madison, WI.

Sveldow, M., C.D. McGillem, and P.E. Anuta, 1976. Experimental Examination of Similarity Measures and Processing Methods used for Image Registration. In the Symposium of Machine Processing of Remotely Sensed Data, Westville, Ind., June, pp. 4A-9.

Theußl, T., 2000. Sampling and Reconstruction in Volume Visualization. M.Sc. Thesis, Vienna University of Technology, Vienna, Austria.

Thevenaz, P., U.E. Ruttimann, and M. Unser, 1998. A Pyramid Approach to Subpixel Registeration Based on Intensity. IEEE Transactions on Image Processing, Vol. 7, No. 1, pp. 27-40.

Vane, G., R.O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen, and W. M. Porter, 1993. The Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). Remote Sensing of Environment. Vol. 44, No.2, pp. 127-143.

Wahl, F.M., 1987. Digital Image Signal Processing. Artech House, Boston.

Warner, W.S., R.W. Graham and R.E. Read, 1996. Small Format Aerial Photography.Whittles Publishing, Scotland, UK.

Watson Industries Inc., 1995.Attitude and Heading Reference System: AHRS-BA303 Owner's Manual. Watson Industries, Inc.

Watson, K., S.H. Miller and D.L. Sawatzky, 1982. Registration of Heat Capacity Mapping Mission day and night images. Photogrammetric Engineering and Remote Sensing, Vol. 48, pp 263-268.

Westin, T., 1990. Precision Rectification of SPOT Imagery. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 2, pp. 247-253.

Wie, P.V., M. Stein, 1977. A Landsat Digital Image Rectification System. IEEE Transactions on Geoscience Electronics. GE-15, pp130-137.

Wolberg, G., 1990. Digital Image Warping. IEEE Computer Society Press, Los Alamitos, CA.

Wolf, P. R., 1983. Elements of Photogrammetry, 2nd Edition. McGraw-Hill, Inc, New York.

Wolf, P.R. and C. D. Ghilani, 1997. Adjustment Computations: Statistics and Least Squares in Surveying and GIS. John Wiley & Sons, Inc., New York.

Wolf, P.R., and B. Dewitt, 2000. Elements of Photogrammetry: with Applications in GIS, 3rd ed. McGraw-Hill Companies, Inc., Boston.

BIOGRAPHICAL SKETCH

Amr Abd-Elrahman was born in Cairo, Egypt, on February 1st, 1969. He received his B.Sc. degree in civil engineering (Public Works Department), from Ain Shams University, Cairo, Egypt, in May, 1990. From 1991 to 1995, he worked as instructor at the Surveying Program, Public Works Department, Ain Shams University. He received his M.Sc. from the same department in GIS and digital mapping in January 1995. In March, 1995, he was promoted to Assistant Lecturer position at the Public Works department, Ain Shams University. He received a governmental scholarship to continue his graduate study in the remote sensing field in the United States in 1996. He joined the Geomatics Program, Civil and Coastal Engineering Department, University of Florida, in January 1997. He was accepted for Ph.D. candidacy at the University of Florida in August 1999.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Scot E. Smith
Associate Professor of Civil and Coastal Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Bon Dewitt
Associate Professor of Civil and Coastal Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Grenville Barnes
Associate Professor of Civil and Coastal Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Abdelsalam Helal
Associate Professor Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Edward Walsh
Professor of Aerospace Engineering, Mechanics and Engineering Science

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

December 2001

[illegible]
Pramod Khargonekar
Dean, College of Engineering

Winfred Phillips
Dean, Graduate School